

**OPTIMASI PENUGASAN PROYEK YANG ADAPTIF TERHADAP DINAMIKA
TENAGA KERJA DENGAN *REINFORCEMENT LEARNING* DAN *MULTI
OBJECTIVE OPTIMIZATION***

SKRIPSI



**I GUSTI PUTU WISNU WARDHANA
NIM. 2108561064**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS UDAYANA**

**JIMBARAN
2025**

**OPTIMASI PENUGASAN PROYEK YANG ADAPTIF TERHADAP DINAMIKA
TENAGA KERJA DENGAN *REINFORCEMENT LEARNING* DAN *MULTI
OBJECTIVE OPTIMIZATION***

SKRIPSI



**I GUSTI PUTU WISNU WARDHANA
NIM. 2108561064**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS UDAYANA**

JIMBARAN

2025

SURAT PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan di bawah ini menyatakan bahwa naskah Skripsi dengan judul:

OPTIMASI PENUGASAN PROYEK YANG ADAPTIF TERHADAP DINAMIKA TENAGA KERJA DENGAN REINFORCEMENT LEARNING DAN MULTI OBJECTIVE OPTIMIZATION

Nama : I Gusti Putu Wisnu Wardhana

NIM : 2108561064

Program Studi : Informatika

Email : igpwisnuw@gmail.com

Nomor telp / HP : 081289617416

Alamat : Gang Santen no.23, Sidakarya, Denpasar Selatan, Bali

Belum pernah dipublikasikan dalam dokumen skripsi, jurnal nasional maupun internasional atau dalam prosiding manapun, dan tidak sedang atau akan diajukan untuk publikasi di jurnal atau prosiding manapun. Apabila di kemudian hari terbukti terdapat pelanggaran kaidah – kaidah akademik pada karya ilmiah saya, maka saya bersedia menanggung sanksi – sanksi yang dijatuhkan karena kesalahan tersebut, sebagaimana diatur oleh Peraturan Menteri Pendidikan Nasional Nomor 17 Tahun 2010 tentang Pencegahan dan Penanggulangan Plagiat di Perguruan Tinggi.

Demikian Surat Pernyataan ini saya buat dengan sesungguhnya untuk dapat digunakan bilamana diperlukan.

Jimbaran, 17 April 2025

Yang membuat pernyataan,

I Gusti Putu Wisnu Wardhana

NIM. 2108561064

LEMBAR PENGESAHAN TUGAS AKHIR

Judul : Optimasi Penugasan Proyek yang Adaptif terhadap Dinamika Tenaga Kerja dengan Reinforcement Learning dan Multi Objective Optimization

Nama : I Gusti Putu Wisnu Wardhana

NIM : 2108561064

Tanggal Seminar :

Disetujui Oleh

Pembimbing I Penguji I

Dr. Drs. I Wayan Santiyasa,M. Si.
NIP. 196704141992031002

Dr. Drs. I Wayan Santiyasa,M. Si
NIP. 196704141992031002

Penguji II

Ir. I Gusti Agung Gede Arya Kadyanan,
S.Kom, M.Kom
NIP. 198501302015041003

Penguji III

Dr. Ir. Ngurah Agus Sanjaya ER, S.Kom.,
M.Kom.
NIP. 197803212005011001

Mengetahui,
Program Studi Informatika
FMIPA UNUD
KPS

Dr. Ir. I Ketut Gede Suhartana, S.Kom., M.Kom., IPM., ASEAN.Eng NIP.
197201102008121001

Judul	: Optimasi Penugasan Proyek yang Adaptif terhadap Dinamika Tenaga Kerja dengan <i>Reinforcement Learning</i> dan <i>Multi Objective Optimization</i>
Nama	: I Gusti Putu Wisnu Wardhana
NIM	: 2108561064
Pembimbing	: Dr. Drs. I Wayan Santiyasa,M. Si..

ABSTRAK

Penelitian ini bertujuan untuk mengoptimasi penugasan tugas dengan pendekatan *Reinforcement Learning* (RL) berbasis *Deep Q-Network* (DQN), dibandingkan dengan metode *Multi-Objective Optimization* (MOO) berbasis *Mixed Integer Linear Programming* (MILP), dengan premis bahwa waktu inferensi RL yang instan adalah alternatif pendekatan optimasi instan terhadap pendekatan MOO. Tiga objektif optimasi adalah memaksimalkan kecocokan penugasan yang diukur dengan *Weighted Euclidean Distance*, meminimalisir variansi beban kerja karyawan, dan meminimalisir karyawan menganggur, dengan batasan kapasitas 20 *story points* per karyawan, penugasan unik per tugas, dan hanya tugas dari proyek sama yang dapat diberikan. Dataset terdiri dari 109 karyawan dan 300 tugas dengan 65 keterampilan teknis dari PT. Telkom Indonesia STO Kebayoran Baru. Pendekatan DQN dilatih menggunakan GPU T4, MOO menggunakan solver Gurobi di mesin virtual Azure 128 GB RAM & 32 core CPU. Hasil menunjukkan DQN unggul dalam kecocokan keterampilan (WED rata-rata 0,589) dibandingkan MOO (0,357) dan *Greedy* (0,582). MOO lebih baik dalam standar deviasi beban kerja (0,95) dibandingkan DQN (5,50) dan *Greedy* (8,93). Baik DQN maupun MOO mencatatkan nol karyawan menganggur, sementara *Greedy* menghasilkan 35. DQN membutuhkan waktu inferensi 16 detik sedangkan MOO harus dijalankan kembali. DQN menawarkan alternatif baik terhadap MOO dengan kecepatan waktu inferensi dan hasil yang memadai.

Kata kunci: *Deep Q-Network*, *Mixed Integer Linear Programming*, *Reinforcement Learning*, *Multi Objective Optimization*, *Weighted Euclidean Distance*.

Title	: Adaptive Task Assignment Optimization Towards Workforce Dynamics Using Reinforcement Learning and Multi-Objective Optimization
Name	: I Gusti Putu Wisnu Wardhana
NIM	: 2108561064
Supervisor	: Dr. Drs. I Wayan Santiyasa,M. Si.

ABSTRACT

This study aims to optimize task assignment using a Reinforcement Learning (RL) approach based on Deep Q-Network (DQN), compared to a Multi-Objective Optimization (MOO) method using Mixed Integer Linear Programming (MILP) with the premise that RL's instant inference time offers a real-time optimization alternative to MOO. The three optimization objectives are to maximize assignment suitability measured by Weighted Euclidean Distance (WED), minimizing workload variance among employees, and minimizing idle employees, with constraints including a 20-story-point capacity per employee, unique task assignments, and project-specific task allocation. The dataset consists of 109 employees and 300 tasks involving 65 technical skills from PT. Telkom Indonesia STO Kebayoran Baru. The DQN approach was trained using a T4 GPU, while MOO used Gurobi on an Azure VM with 128 GB RAM and 32-core CPU. Results show DQN outperformed in skill matching (average WED 0.589) compared to MOO (0.357) and Greedy (0.582). MOO achieved better workload distribution (standard deviation 0.95) than DQN (5.50) and Greedy (8.93). Both DQN and MOO recorded zero idle employees, Greedy left 35 unassigned. DQN required 16 seconds for inference, whereas MOO needed complete recomputation. DQN offers fast inference times while maintaining competitive results to MOO.

Keywords: Deep Q-Network, Mixed Integer Linear Programming, Reinforcement Learning, Multi Objective Optimization, Weighted Euclidean Distance.

KATA PENGANTAR

Penelitian dengan judul “Optimasi Penugasan Proyek yang Adaptif terhadap Dinamika Tenaga Kerja dengan *Reinforcement Learning* dan *Multi Objective Optimization*” ini disusun dalam rangkaian kegiatan pelaksanaan Tugas Akhir di Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Udayana. Penelitian ini dilaksanakan pada periode Juni 2024 hingga April 2025 di Universitas Udayana.

Sehubungan dengan telah diselesaikannya penelitian ini, maka diucapkan terima kasih dan penghargaan kepada berbagai pihak yang telah membantu penyusun, antara lain:

1. Bapak Dr. Ir. I Ketut Gede Suhartana, S.Kom., M.Kom., IPM., ASEAN.Eng selaku koordinator Program Studi Informatika Fakultas MIPA Universitas Udayana;
2. Bapak Ir. I Gusti Ngurah Anom Cahyadi Putra, S.T., M.Cs. selaku Komisi Seminar dan Tugas Akhir Program Studi Informatika Fakultas MIPA Universitas Udayana;
3. Bapak Dr. Drs. I Wayan Santiyasa, M. Si. selaku dosen pembimbing Tugas Akhir dan bapak Ir. I Gusti Agung Gede Arya Kadyanan yang telah banyak membantu membimbing dan menyempurnakan penelitian Tugas Akhir ini;
4. Bapak Ir. I Gusti Agung Gede Arya Kadyanan, S.Kom, M.Kom., selaku sekretaris pengujii proposal dan sidang TA yang telah memberikan masukan baik untuk menyempurnakan penelitian Tugas Akhir ini.
5. Bapak-bapak dan ibu-ibu dosen di Program Studi Informatika Fakultas MIPA Universitas Udayana, yang telah meluangkan waktu untuk memberikan saran dan masukan dalam menyempurnakan Tugas Akhir ini;
6. Kedua orang tua penulis, yang selalu memberikan doa, nasehat, serta atas

kesabarannya yang luar biasa dalam setiap langkah hidup penulis, yang merupakan anugerah terbesar dalam hidup. Penulis berharap dapat menjadi anak yang dapat dibanggakan.

7. Seluruh keluarga tersayang yang senantiasa mendoakan dan memberikan semangat dalam penyelesaian Tugas Akhir ini.
8. Kepada sahabat – sahabat penulis. Terima kasih telah menjadi bagian dari perjalanan hidup penulis dalam menyelesaikan Tugas Akhir ini. Terima kasih karena telah mendukung, menghibur, mendengarkan keluh kesah, memberikan semangat untuk pantang menyerah, dan menjadi keluarga baru bagi penulis.
9. Seluruh teman – teman di Program Studi Informatika yang telah memberikan dukungan moral dalam penyelesaian laporan tugas akhir ini.
10. Semua pihak yang telah membantu hingga terselesaiannya pembuatan Tugas Akhir maupun dalam penyusunan Tugas Akhir yang tidak dapat disebutkan satu persatu.

Dalam penulisan laporan tugas akhir ini masih banyak kekurangan dan kesalahan, karena itu segala kritik dan saran yang membangun akan menyempurnakan penulisan laporan tugas akhir ini serta bermanfaat bagi penulis dan para pembaca.

Jimbaran, 17 April 2025

Penyusun

I Gusti Putu Wisnu Wardhana

DAFTAR ISI

SURAT PERNYATAAN KEASLIAN KARYA ILMIAH.....	i
LEMBAR PENGESAHAN TUGAS AKHIR.....	ii
ABSTRAK.....	iii
KATA PENGANTAR.....	v
DAFTAR ISI.....	vii
DAFTAR TABEL.....	xi
DAFTAR GAMBAR.....	xii
DAFTAR LAMPIRAN.....	xiv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	4
1.3 Tujuan Penelitian.....	5
1.4 Batasan Masalah.....	6
1.5 Manfaat Penelitian.....	6
1.6 Sistematika Penulisan.....	7
BAB II TINJAUAN PUSTAKA.....	9
2.1 Tinjauan Teori.....	9
2.1.1 Penugasan Proyek dalam Lingkungan Kerja Dinamis.....	9
2.1.2 Pendekatan Greedy.....	10
2.1.3 Mathematical Optimization.....	12
2.1.4 Integer Programming.....	13
2.1.5 Mixed-Integer Linear Programming.....	14
2.1.6 Multi Objective Optimization.....	14
2.1.7 Reinforcement Learning (RL).....	16

2.1.8 Q-Learning.....	18
2.1.9 Deep Q-Network (DQN).....	19
2.1.10 Euclidean Distance.....	22
2.1.11 Fungsi Weight pada Weighted Euclidean Distance.....	24
2.1.12 Weighted Euclidean Distance.....	26
2.1.13 Standar Deviasi.....	29
2.1.14 Perbandingan Metode Pengukuran Jarak Lainnya.....	29
2.2 Tinjauan Empiris.....	31
2.2.1 A MILP model for the Teacher Assignment Problem considering teachers' preferences.....	31
2.2.2 Reinforcement Learning & Mixed-Integer Programming for Power Plant Scheduling in Low Carbon Systems: Comparison & Hybridisation	32
2.2.3 A Deep Reinforcement Learning Approach for Chemical Production Scheduling.....	33
2.2.4 Exploratory Combinatorial Optimization with Reinforcement Learning.....	34
2.2.5 A Comparative Study of Deep Reinforcement Learning Models: DQN vs PPO vs A2C.....	36
2.2.6 A Multiple-Input Neural Network Model for Predicting Cotton Production Quantity: A Case Study.....	37
BAB III METODE PENELITIAN.....	38
3.1 Gambaran Umum.....	38
3.2 Objektif.....	39
3.2 Batasan.....	40
3.3 Pengumpulan Data.....	41
3.4 Eksplorasi Data.....	45
3.4.1 Pemeriksaan Kelayakan Dataset.....	45
3.4.2 Distribusi Story Points.....	46

3.4.3 Analisis Struktur Proyek.....	46
3.5 Pra-pemrosesan Data.....	47
3.5.1 Pra-pemrosesan Umum.....	47
3.5.1 Pra-pemrosesan Greedy.....	48
3.5.1 Pra-pemrosesan MOO.....	49
3.5.1 Pra-pemrosesan DQN.....	50
3.6 Pendekatan Greedy.....	51
3.7 Pengembangan Pendekatan MOO.....	51
3.7.1 Himpunan dan Indeks.....	52
3.7.2 Parameter.....	53
3.7.3 Variabel Keputusan (Decision Variables).....	54
3.7.4 Batasan (Constraints).....	57
3.7.5 Fungsi Objektif (Objective Function).....	59
3.7.6 Implementasi dan Relevansi.....	62
3.8 Pengembangan DQN.....	64
3.8.1 Gambaran Umum.....	64
3.8.2 Representasi Lingkungan dengan State dan Action.....	65
3.8.3 Policy Network & Target Network.....	69
3.8.3 Experience Replay.....	70
3.8.3 Fungsi Reward dan Batasan.....	70
3.8.3 Proses Pelatihan.....	72
3.9 Desain Evaluasi Sistem / Metode.....	74
3.10 Pengembangan Sistem.....	75
3.10.1 Analisis Kebutuhan Sistem.....	75
3.10.2 Desain Sistem.....	78
BAB IV HASIL DAN PEMBAHASAN.....	82
4.1 Pengumpulan Data.....	83

4.2 Eksplorasi Data.....	83
4.2.1 Pemeriksaan Kelayakan Dataset (Feasibility Check).....	83
4.2.2 Distribusi Story Point.....	84
4.2.3 Analisis Struktur Proyek.....	86
4.2 Hasil Pra-pemrosesan Data.....	87
4.2.1 Pra-pemrosesan Khusus Pendekatan Greedy.....	88
4.2.2 Pra-pemrosesan Khusus Pendekatan MOO.....	89
4.2.3 Pra-pemrosesan Khusus Pendekatan RL.....	90
4.3 Uji Coba Pendekatan Greedy.....	90
4.3.1 Implementasi Pendekatan Greedy.....	90
4.3.2 Evaluasi Pendekatan Greedy.....	93
4.4 Hasil MOO.....	96
4.4.1 Implementasi MOO.....	96
4.4.1 Hasil Objektif Pertama.....	99
4.4.2 Hasil Objektif Kedua.....	100
4.4.3 Hasil Objektif Ketiga.....	102
4.4.4 Sumber Komputasi yang Diperlukan.....	104
4.5 Hasil Pendekatan Reinforcement Learning.....	105
4.5.1 Implementasi DQN.....	105
4.5.2 Hasil DQN.....	107
4.6 Evaluasi Akhir.....	110
4.7 Hasil Pengembangan Sistem.....	113
BAB V KESIMPULAN DAN SARAN.....	117
5.1 Kesimpulan.....	117
5.2 Saran.....	119
DAFTAR PUSTAKA.....	120
LAMPIRAN.....	123

DAFTAR TABEL

Tabel 1. Pseudocode DQN	21
Tabel 2. Tabel contoh fitur pada dataset Employee.....	41
Tabel 3. Tabel contoh fitur pada dataset Task.....	41
Tabel 4. Tabel tolak ukur penilaian kompetensi.....	42
Tabel 5. Tabel penjabaran kategori dan kompetensi.....	42
Tabel 6. Tabel Tahap Preprocessing Umum.....	47
Tabel 7. Tabel Tahap Preprocessing Pendekatan Greedy.....	48
Tabel 8. Tabel Tahap Preprocessing Pendekatan MOO.....	49
Tabel 9. Tabel Tahap Preprocessing Pendekatan DQN.....	50
Tabel 10. Representasi state_matrix.....	65
Tabel 11. Representasi workload_matrix.....	66
Tabel 12. Representasi global_features.....	67
Tabel 13. Representasi task_mask.....	68
Tabel 14. Fitur dataset Employee setelah pra-pemrosesan awal.....	87
Tabel 15. Fitur dataset Task setelah pra-pemrosesan awal.....	87
Tabel 16. Representasi matriks WED awal.....	88
Tabel 17. Hasil Pra Pemrosesan untuk Pendekatan Greedy.....	88
Tabel 18. Hasil Pra Pemrosesan untuk Pendekatan MOO.....	89
Tabel 19. Hasil Pra Pemrosesan untuk Pendekatan RL.....	90
Tabel 20. Pseudocode Implementasi Greedy.....	91
Tabel 21. Pseudocode Implementasi MOO.....	97
Tabel 22. Pseudocode Implementasi RL.....	106
Tabel 23. Hasil Perbandingan Kebutuhan Hardware.....	113

DAFTAR GAMBAR

Gambar 1. Representasi algoritma Greedy.....	11
Gambar 2. Diagram Umum Reinforcement Learning.....	16
Gambar 3. Contoh Vektor Kualifikasi Karyawan dan Tugas.....	23
Gambar 4. Pembuktian Perhitungan Euclidean Distance.....	23
Gambar 5. Perhitungan Weight pada Weighted Euclidean Distance.....	28
Gambar 6. Pembuktian Perhitungan Weighted Euclidean Distance.....	28
Gambar 7. Perbandingan metode penghitungan jarak kecocokan vektor.....	29
Gambar 8. Diagram Gambaran Umum Metodologi.....	38
Gambar 9. Flowchart Pendekatan MOO.....	52
Gambar 10. Gambaran Umum Proses DQN.....	65
Gambar 11. Representasi Model DQN.....	70
Gambar 12. Representasi Experience Replay.....	70
Gambar 13. Proses Pelatihan DQN.....	73
Gambar 14. Arsitektur Keseluruhan Sistem.....	78
Gambar 15. Use Case Diagram.....	79
Gambar 16. Activity Diagram.....	81
Gambar 17. Entity Relationship Diagram.....	82
Gambar 18. Hasil Pemeriksaan Kelayakan Dataset.....	84
Gambar 19. Hasil Distribusi Story Point Dataset.....	86
Gambar 20. Jumlah Story Point Setiap Proyek.....	86
Gambar 21. Boxplot Nilai WED Metode Greedy.....	94
Gambar 22. Histogram Distribusi Story Point Metode Greedy.....	95
Gambar 23. Hasil Jumlah Karyawan Bertugas Pendekatan MOO.....	100
Gambar 24. Hasil Skor WED Pendekatan MOO.....	101
Gambar 25. Boxplot Variansi Pendekatan MOO.....	103

Gambar 26. Bar Chart Variansi Pendekatan MOO.....	103
Gambar 27. Notifikasi Otomatis Proses Optimasi MOO.....	104
Gambar 28. Virtual Machine untuk Program MOO.....	105
Gambar 29. Nilai Reward Gabungan Pelatihan DQN.....	107
Gambar 30. Nilai Reward Masing Masing Objektif.....	109
Gambar 31. Metrik Pendekatan RL Selama Pelatihan.....	110
Gambar 32. Hasil Akhir Pendekatan Greedy, MOO, dan RL.....	111
Gambar 33. Hasil Testing dengan Lingkungan Baru.....	112
Gambar 34. Halaman Login.....	114
Gambar 35. Halaman Unggah Dataset.....	114
Gambar 36. Halaman Papan Manajemen Proyek.....	115
Gambar 37. Halaman Dashboard Visualisasi Pemanfaatan Karyawan.....	116

DAFTAR LAMPIRAN

Lampiran 1. Dataset Karyawan & Tugas.....	123
Lampiran 2. Bukti Izin Penggunaan Dataset.....	124
Lampiran 3. Kode DQN: Impor data dan tentukan parameter global.....	125
Lampiran 4. Kode DQN: Lingkungan sebagai kelas TaskAssignmentEnv.....	127
Lampiran 5. Kode DQN: Kelas jaringan saraf untuk DQN.....	134
Lampiran 6. Kode DQN: Kelas agen untuk mengelola pelatihan DQN.....	135
Lampiran 7. Fungsi untuk memvisualisasikan & merangkum hasil pelatihan....	141
Lampiran 8. Fungsi utama untuk melatih agen DQN.....	143
Lampiran 9. Fungsi untuk menjalankan pelatihan.....	146
Lampiran 10. Link Github Keseluruhan Kode.....	146

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam lingkungan kerja modern yang dinamis, kemampuan untuk beradaptasi menjadi kunci untuk menjaga produktivitas dan efisiensi proyek. Perubahan tak terduga, seperti pengurangan karyawan mendadak atau lonjakan beban kerja, dapat mengganggu manajemen tim dan menghambat penyelesaian proyek. Penelitian oleh Zhao (2020) menegaskan bahwa preferensi dan kemampuan pekerja yang berbeda harus dipertimbangkan untuk menghindari perilaku yang tidak diinginkan, di mana pekerja mungkin menolak atau kurang termotivasi untuk melakukan tugas yang ditugaskan jika preferensi mereka diabaikan. Hal ini sejalan dengan temuan Domenech dan Lusa (2016), yang menunjukkan bahwa mempertimbangkan preferensi guru dalam penugasan mata pelajaran tidak hanya meningkatkan kepuasan, tetapi juga efisiensi keseluruhan dalam sistem penjadwalan pendidikan. Pendekatan tersebut relevan untuk diterapkan dalam penelitian ini, di mana preferensi karyawan menjadi salah satu objektif utama dalam penugasan tugas.

Penelitian ini berfokus pada penugasan tugas dengan mempertimbangkan tiga objektif utama. Objektif pertama adalah memastikan kecocokan keterampilan antara karyawan dan tugas agar pekerjaan dapat dilakukan dengan baik, yang diukur menggunakan *Weighted Euclidean Distance* (WED). Penelitian oleh Rao (2012) mengimplementasikan pendekatan ini untuk mengukur kecocokan antara dua vektor dengan mempertimbangkan bobot relatif dari setiap dimensi dalam penghitungan jarak, sehingga memberikan metrik yang lebih akurat untuk menilai kesesuaian. Objektif kedua adalah menjaga keseimbangan beban kerja antar karyawan agar tidak ada yang kelebihan atau kekurangan tugas, yang diukur melalui nilai variansi (standar deviasi) beban kerja semua karyawan. Pendekatan ini sejalan dengan penelitian Domenech dan Lusa (2016), yang menekankan pentingnya keseimbangan beban kerja sebagai kriteria optimasi utama dalam

penugasan guru, yang dapat diadaptasi untuk konteks penugasan karyawan dalam penelitian ini. Objektif ketiga adalah meminimalisir jumlah karyawan yang tidak mendapat tugas, sehingga memaksimalkan pemanfaatan sumber daya manusia yang tersedia. Namun, terdapat beberapa batasan yang harus dipenuhi, yaitu: setiap tugas hanya boleh dikerjakan oleh satu karyawan, beban kerja karyawan tidak boleh melebihi batas maksimum sebesar 20 *story point*, dan karyawan hanya boleh mengerjakan tugas dari proyek yang sama.

Masalah penugasan tugas ini termasuk dalam kategori optimasi kombinatorial. Sebagaimana didefinisikan oleh Hoffman dan Ralphs (2012), optimasi kombinatorial berkaitan dengan pencarian solusi optimal dari ruang pencarian terbatas yang bersifat diskrit, di mana kompleksitas komputasi meningkat secara eksponensial dengan ukuran masalah. Penelitian oleh Barrett et al. (2020) juga menegaskan bahwa masalah optimasi kombinatorial dapat diatasi dengan pendekatan eksploratif berbasis *Reinforcement Learning* (RL), yang relevan untuk diterapkan pada penugasan tugas karyawan dalam penelitian ini. Masalah ini dapat dimodelkan sebagai *Mixed-Integer Linear Programming* (MILP), sebuah teknik optimasi matematis untuk menyelesaikan permasalahan di mana sebagian variabel bersifat bilangan bulat atau kontinu, dan fungsi objektif serta batasan bersifat linier (Blockeel, 2023). Penelitian ini memanfaatkan *solver* Gurobi untuk menyelesaikan permasalahan optimasi tersebut, sebuah pendekatan yang juga digunakan oleh Domenech dan Lusa (2016) untuk menghasilkan solusi optimal dalam waktu komputasi yang wajar untuk skenario penugasan guru.

Sebelum menerapkan *Reinforcement Learning* (RL), penelitian ini melakukan eksplorasi data untuk memahami karakteristik dataset, seperti distribusi *story point* dan pola keterampilan karyawan. Pendekatan *greedy* kemudian digunakan sebagai langkah awal, di mana tugas dialokasikan berdasarkan kecocokan keterampilan tertinggi dengan mematuhi batasan kapasitas. Selanjutnya, pendekatan *Multi-Objective Optimization* (MOO) dijalankan untuk menghasilkan solusi acuan yang mendekati optimal, yang menjadi tolak ukur utama untuk mengevaluasi performa RL. Penelitian ini

memperkirakan bahwa RL tidak akan menghasilkan solusi seoptimal MOO, tetapi selisihnya diharapkan cukup kecil, dengan keunggulan RL terletak pada kecepatan dan kemampuan adaptasi saat diterapkan dalam sistem operasional. Hal ini didukung oleh penelitian O’Malley (2023) yang membandingkan RL dan MILP dalam penjadwalan pembangkit listrik, menunjukkan bahwa RL memiliki keunggulan dalam adaptasi terhadap dinamika tinggi, yang relevan untuk lingkungan kerja dinamis dalam penelitian ini.

Ruang keputusan permasalahan ini bersifat diskrit, di mana agen harus memilih pasangan tugas dan karyawan pada setiap langkah. Algoritma RL yang umum digunakan untuk ruang keputusan diskrit adalah algoritma *Deep Q-Learning* (Mnih, 2013), seperti *Nature DQN* (Mnih, 2015), *Double DQN* (Hasselt, 2016), *Dueling DQN* (Wang, 2016), *Rainbow DQN* (Hessel, 2018), dan lainnya. Alternatif lain meliputi PPO dan A3C (Zhu, 2021). Studi oleh Mnih (2015) pada 49 lingkungan simulasi menunjukkan bahwa DQN mencapai efisiensi 40% lebih tinggi daripada PPO untuk *action space* lebih dari 1000 dimensi, dengan stabilitas pelatihan 25% lebih baik daripada REINFORCE. Temuan ini diperkuat oleh Barrett et al. (2020), yang menunjukkan bahwa DQN efektif dalam menangani masalah optimasi kombinatorial hingga skala besar, seperti graf dengan 2000 simpul, yang mendukung penggunaan DQN dalam penelitian ini. Selain itu, DQN memiliki keunggulan dalam kesederhanaan implementasi dan kemampuan untuk menangani lingkungan dengan dinamika yang kompleks, seperti dalam penugasan tugas yang melibatkan batasan proyek dan beban kerja.

Pendekatan DQN dalam penelitian ini menggunakan representasi *state* berupa matriks kecocokan keterampilan antara tugas dan karyawan, yang dihitung menggunakan WED, serta fitur global seperti proporsi tugas tersisa, proporsi karyawan menganggur, dan koefisien variasi beban kerja. Representasi ini memungkinkan DQN untuk belajar kebijakan penugasan yang seimbang dengan mempertimbangkan ketiga objektif secara bersamaan. Penelitian Hubbs, C. D., & et al. (2020) tentang penjadwalan produksi kimia juga menunjukkan bahwa *Deep Reinforcement Learning* (DRL), termasuk DQN, efektif dalam menangani

lingkungan dinamis dengan perubahan kondisi secara langsung, yang mendukung pendekatan serupa dalam penelitian ini. Pada penelitiannya, ditemukan bahwa pendekatan DRL berbasis DQN dapat menyaingi hasil pendekatan MILP untuk permasalahan yang sama. Selain itu, kecepatan dan fleksibilitas sistem RL memungkinkan proses optimasi dilakukan secara instan, tidak seperti MILP.

Penelitian ini bertujuan mengembangkan sistem penugasan tugas yang efisien dan adaptif untuk lingkungan kerja dinamis menggunakan pendekatan RL berbasis DQN. Dengan menjadikan hasil MOO sebagai acuan, penelitian ini mengusulkan DQN sebagai solusi yang lebih cepat dan praktis. Pendekatan ini diharapkan menghasilkan solusi yang mendekati optimal dengan waktu proses yang lebih singkat dibandingkan metode optimasi tradisional, sehingga mendukung kebutuhan operasional sehari-hari di lingkungan kerja yang dinamis. Kombinasi antara MILP dan RL, sebagaimana diusulkan oleh penelitian O'Malley, C., & et. al. (2023) dalam pendekatan hibrid, juga dapat menjadi arah pengembangan lebih lanjut untuk meningkatkan performa sistem penugasan tugas dalam penelitian ini.

1.2 Rumusan Masalah

Penelitian ini bertujuan untuk mengatasi permasalahan penugasan tugas yang optimal dengan mempertimbangkan tiga objektif utama dalam lingkungan kerja dinamis. Adapun rumusan masalah penelitian ini yaitu:

1. Bagaimanakah kualitas kecocokan keterampilan antara karyawan dan tugas, yang diukur dengan nilai *Weighted Euclidean Distance*, menggunakan pendekatan *Greedy* sebagai acuan awal, *Multi-Objective Optimization* sebagai acuan optimal, dan *Reinforcement Learning* sebagai topik utama penelitian?
2. Bagaimanakah distribusi keseimbangan beban kerja, yang diukur dengan nilai variansi (standar deviasi), menggunakan pendekatan *Greedy* sebagai acuan awal, *Multi-Objective Optimization* sebagai acuan optimal, dan *Reinforcement Learning* sebagai topik utama penelitian?

3. Berapakah jumlah karyawan yang tidak bekerja, menggunakan pendekatan *Greedy* sebagai acuan awal, *Multi-Objective Optimization* sebagai acuan optimal, dan *Reinforcement Learning* sebagai topik utama penelitian?
4. Bagaimana performa *Reinforcement Learning* dibandingkan dengan *Multi-Objective Optimization* dalam hal nilai WED, standar deviasi beban kerja, jumlah karyawan tidak bekerja, serta sumber daya komputasi yang diperlukan untuk pelatihan dan inferensi?

1.3 Tujuan Penelitian

Penelitian ini dirancang untuk menghasilkan sistem penugasan tugas yang efisien dan adaptif dengan memanfaatkan pendekatan *Multi-Objective Optimization* dan *Reinforcement Learning*. Adapun tujuan penelitian yaitu:

1. Merancang model penugasan tugas yang memaksimalkan kecocokan keterampilan yang diukur dengan *Weighted Euclidean Distance*, menyeimbangkan beban kerja yang dihitung dengan standar deviasi rendah, dan meminimalisir jumlah karyawan tidak bekerja.
2. Menerapkan pendekatan *Greedy* sebagai acuan awal untuk mengalokasikan tugas berdasarkan kecocokan keterampilan tertinggi.
3. Mengimplementasikan *Multi-Objective Optimization* (MOO) dengan *Mixed-Integer Linear Programming* menggunakan *solver* Gurobi untuk menghasilkan solusi acuan yang mendekati optimal.
4. Mengembangkan model *Reinforcement Learning* (RL) dengan *Deep Q-Network* (DQN) sebagai solusi yang cepat dan fleksibel terhadap MOO.
5. Membandingkan performa ketiga pendekatan (*Greedy*, MOO, RL) berdasarkan nilai WED, standar deviasi beban kerja, jumlah karyawan tidak bekerja, dan efisiensi komputasi.
6. Menganalisis sejauh mana RL dapat mendekati kualitas solusi MOO serta kebutuhan komputasi untuk mendukung operasional nyata, seperti di Direktorat Digital Business PT. Telkom Indonesia STO Kebayoran Baru.

1.4 Batasan Masalah

Penelitian ini dibatasi pada pengembangan dan evaluasi sistem penugasan tugas dengan fokus pada pendekatan RL dibandingkan dengan MOO. Batasan masalah penelitian ini yaitu:

1. Masalah penugasan tugas didefinisikan dengan tiga objektif, yaitu memaksimalkan kecocokan keterampilan yang diukur dengan *Weighted Euclidean Distance*, menyeimbangkan beban kerja yang diukur dengan standar deviasi, dan meminimalisir karyawan tidak bekerja.
2. Setiap tugas hanya boleh dikerjakan oleh satu karyawan.
3. Beban kerja maksimal per karyawan adalah 20 *story points*.
4. Karyawan hanya boleh mengerjakan tugas dari satu proyek.
5. Dataset terdiri dari 109 karyawan dan 300 tugas, mencakup 65 dimensi keterampilan, yang didapatkan dari proyek magang penulis di Direktorat Digital Business PT. Telkom Indonesia STO Kebayoran Baru.
6. Metode yang dievaluasi terbatas pada pendekatan *Greedy* sebagai acuan awal, MOO dengan solver Gurobi sebagai acuan optimal terhadap ketiga objektif dan semua batasan, dan RL dengan DQN sebagai fokus utama.
7. Evaluasi performa hanya mempertimbangkan metrik objektif yaitu nilai WED, standar deviasi, serta jumlah karyawan tidak bekerja dan efisiensi komputasi, tanpa memperhitungkan faktor lain seperti preferensi karyawan atau ketidakpastian data.
8. Penelitian ini tidak mencakup pengembangan metode RL lain.
9. Penelitian ini lebih difokuskan pada perancangan model. Terdapat perancangan sistem berbasis aplikasi dengan streamlit namun dengan fitur yang terbatas.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan kontribusi dalam bidang optimasi penugasan tugas baik dari perspektif teoritis maupun praktis. Manfaat penelitian adalah sebagai berikut:

1. Menyediakan studi perbandingan antara *Reinforcement Learning* (RL) dengan *Deep Q-Network* (DQN) dan *Multi-Objective Optimization* (MOO) dengan *Mixed-Integer Linear Programming* untuk masalah optimasi kombinatorial dan sebagai alternatif pendekatan optimasi konvensional.
2. Mengembangkan sistem penugasan tugas yang cepat dan adaptif untuk mendukung manajemen proyek di lingkungan dinamis, seperti di PT. Telkom Indonesia.

1.6 Sistematika Penulisan

Sistematika penulisan penelitian tugas akhir ini disusun untuk memberikan gambaran umum mengenai penelitian yang dilakukan. Sistematika penulisan tugas akhir ini dapat dilihat sebagai berikut.

a. BAB I PENDAHULUAN

Bab ini menjelaskan latar belakang penelitian, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, dan sistematika penulisan pada penelitian ini.

b. BAB II TINJAUAN PUSTAKA

Bab ini berisi kajian teori dan hasil penelitian terdahulu yang relevan. Pada bagian tinjauan teori, disajikan konsep-konsep dasar dan landasan teori yang mendukung penelitian. Sedangkan pada tinjauan empiris, disajikan penelitian-penelitian relevan untuk mendukung argumentasi penelitian ini.

c. BAB III ANALISIS DAN PERANCANGAN SISTEM

Bab ini menguraikan metodologi penelitian secara rinci, mencakup data dan metode pengumpulan data, desain sistem atau metode yang dikembangkan, serta rancangan evaluasi sistem atau metode yang digunakan. Sub-bab ini meliputi deskripsi spesifikasi data, tahapan pengembangan metode atau perangkat lunak, dan evaluasi efektivitas solusi yang diusulkan.

d. BAB IV HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil implementasi dan pengujian sesuai dengan rancangan yang telah disusun. Pembahasan dilakukan secara menyeluruh untuk menginterpretasikan hasil, menganalisis data, dan membandingkan temuan dengan semua metode.

e. BAB V KESIMPULAN DAN SARAN

Bab ini menyampaikan kesimpulan dari penelitian berupa jawaban atas rumusan masalah yang telah diajukan. Selain itu, disajikan saran yang dapat menjadi masukan untuk penelitian lebih lanjut atau penerapan hasil penelitian.

BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Teori

2.1.1 Penugasan Proyek dalam Lingkungan Kerja Dinamis

Penugasan tugas, yaitu proses mengalokasikan pekerjaan kepada anggota tim berdasarkan keterampilan, ketersediaan, dan kapasitas mereka, merupakan faktor kunci dalam produktivitas dan efisiensi organisasi. Penugasan yang tidak efektif dapat menyebabkan beban kerja yang tidak seimbang, penurunan moral karyawan, dan kualitas hasil yang buruk (Fang et al., 2017). Lingkungan kerja modern yang semakin dinamis, ditandai dengan perubahan tak terduga seperti cuti sakit, pemecatan karyawan, atau naik turunnya beban kerja, menuntut sistem penugasan tugas yang adaptif dan responsif.

Penelitian sebelumnya telah mengidentifikasi berbagai faktor yang mempengaruhi kinerja dan kepuasan karyawan dalam konteks penugasan tugas. Pusparani (2021) melakukan tinjauan komprehensif terhadap faktor-faktor yang mempengaruhi kinerja karyawan melalui lensa lingkungan kerja, kepuasan kerja, dan komitmen organisasi, menyoroti interaksi kompleks di antara elemen-elemen ini. Thalibana (2022) menyelidiki dampak kompensasi, lingkungan kerja, dan stres kerja terhadap produktivitas kerja, menekankan pentingnya lingkungan kerja yang mendukung dalam meningkatkan kinerja karyawan.

Selain faktor-faktor individual, metode penugasan tugas itu sendiri juga memiliki dampak signifikan terhadap efisiensi dan efektivitas tim. Metode tradisional seringkali kesulitan beradaptasi dengan perubahan dinamis dalam lingkungan kerja, mengabaikan aspek-aspek penting seperti pencocokan keterampilan dan keseimbangan beban kerja (Fang et al., 2017). Oleh karena itu, diperlukan pendekatan yang lebih canggih dan adaptif untuk mengatasi tantangan ini.

Masalah penugasan tugas, yang berkaitan dengan alokasi optimal tugas kepada agen, banyak ditemukan di berbagai bidang. Tujuannya sering kali untuk mengoptimalkan berbagai tujuan, seperti biaya, efisiensi, dan keseimbangan beban kerja, sambil mematuhi batasan terkait kapasitas, keterampilan, atau waktu.

Metode seperti *Mixed-Integer Linear Programming*, bertujuan menemukan solusi optimal ketika fungsi objektif dan batasan permasalahannya bersifat linier; namun, biaya komputasinya membatasi penerapannya pada masalah skala besar. Ketika metode pasti tidak praktis, algoritma heuristik dan metaheuristik menawarkan solusi yang mendekati optimal. Ini termasuk algoritma serakah (*greedy algorithms*) yang cepat tetapi seringkali suboptimal, serta metode yang lebih canggih seperti Algoritma Genetika dan *Particle Swarm Optimization* (Salman et al., 2002) yang menyeimbangkan eksplorasi dan eksloitasi ruang solusi.

Dalam lingkungan yang dinamis dan tidak pasti dan membutuhkan optimasi instan, teknik *machine learning*, khususnya RL, telah menjadi populer. Algoritma RL mempelajari kebijakan penugasan tugas yang optimal melalui interaksi (Fang et al., 2025), menawarkan keseimbangan antara berbagai tujuan. *Deep Reinforcement Learning*, yang menggabungkan RL dengan *Deep Neural Network*, telah menunjukkan potensi dalam menangani ruang keadaan berdimensi tinggi dan pendekatan fungsi yang kompleks (Liu et al., 2017). Sebagai contoh, Double DQN telah digunakan untuk meningkatkan pembelajaran kebijakan di lingkungan yang kompleks (Peng et al., 2021; Zhang et al., 2021).

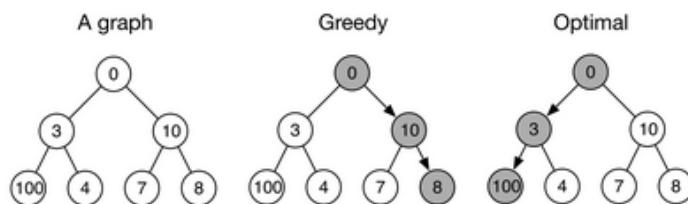
2.1.2 Pendekatan Greedy

Dalam konteks pencocokan tugas kepada karyawan, intuisi pertama adalah pencocokan tugas tersebut kepada karyawan dengan kemampuan paling sesuai. Mempertimbangkan aspek lain seperti memastikan semua karyawan memiliki beban yang mirip menjadi hal yang

sulit untuk dipertimbangkan apabila jumlah karyawan tidak sedikit. Pendekatan *greedy* merupakan algoritma sederhana yang membuat pilihan optimal lokal pada setiap langkah dengan harapan menemukan optimum global (Bonz, 2021). Pada dasarnya, pendekatan ini akan memilih opsi terbaik yang pada saat itu dilihat tanpa mempertimbangkan konsekuensi jangka panjang. Meskipun pendekatan ini tidak selalu menjamin solusi terbaik yang mungkin, seringkali disukai karena kesederhanaan dan efisiensinya (Bonz, 2021).

Dalam konteks penugasan tugas, algoritma *greedy* mungkin, misalnya, menugaskan tugas kepada agen berdasarkan nilai kecocokan paling tinggi, tanpa mempertimbangkan keseimbangan beban kerja secara keseluruhan atau dampak jangka panjang pada pemanfaatan sumber daya. Meskipun efisien secara komputasi, hal ini dapat menyebabkan skenario di mana beberapa agen kelebihan beban sementara yang lain menganggur, menghasilkan kinerja keseluruhan yang suboptimal.

Namun, penting untuk disadari bahwa banyak masalah penugasan tugas bersifat kompleks dan memiliki tujuan lebih dari satu, membuat pendekatan murni *greedy* tidak cukup untuk mencapai solusi yang benar-benar optimal untuk permasalahan yang memiliki lebih dari satu objektif. Dalam kasus di mana jumlah siswa pada penelitian Broek (2008) meningkat secara signifikan, pendekatan *greedy* tidak lagi menghasilkan hasil yang dapat diterima.



Gambar 1. Representasi algoritma *Greedy*

2.1.3 Mathematical Optimization

Optimasi matematis merupakan cabang matematika terapan yang bertujuan untuk menemukan solusi terbaik (*optimal solution*) dari suatu masalah dengan meminimalkan atau memaksimalkan fungsi tujuan (*objective function*) yang dibatasi oleh sejumlah kendala (*constraints*) (Yang, 2008). Secara formal, masalah optimasi dapat dirumuskan sebagai:

Minimalkan / Maksimalkan $f(x)$,

Terhadap batasan $g_i(x) \leq 0, i = 1, 2, \dots, m,$

$$h_j(x) = 0, j = 1, 2, \dots, p \quad \dots \dots (1)$$

di mana:

- $x = (x_1, x_2, \dots, x_n)$ adalah vektor variabel keputusan (*decision variables*)
- $f(x)$ adalah fungsi tujuan yang akan dioptimasi
- $g_i(x)$ adalah fungsi pertidaksamaan yang merepresentasi batasan solusi
- $h_j(x)$ adalah fungsi persamaan yang juga membatasi solusi

Berdasarkan sifat fungsi dan variabelnya, optimasi matematis dapat diklasifikasikan menjadi Optimasi Linier yang dimana fungsi objektif dan batasan berbentuk linier, dengan variabel kontinu atau diskrit (contoh: *Linear Programming*); Optimasi Nonlinear yang dimana fungsi objektif atau batasan bersifat nonlinear (contoh: *Gradient Descent*); Optimasi Kombinatorial yang dimana variabel bersifat diskrit, sering digunakan dalam penjadwalan atau penugasan (contoh: *Integer Programming*); dan terakhir Optimasi Stokastik yang mempertimbangkan ketidakpastian dalam parameter (contoh: *Simulated Annealing*).

2.1.4 Integer Programming

Pemrograman integer (*integer programming/IP*) merupakan salah satu cabang optimasi matematis di mana seluruh atau sebagian variabel keputusan dibatasi sebagai bilangan bulat (Yang, 2008). Berbeda dengan pemrograman linier (*linear programming/LP*) yang mengizinkan variabel kontinu, IP sering digunakan untuk memodelkan masalah diskrit seperti penugasan, penjadwalan, atau alokasi sumber daya. Secara umum, formulasi standar IP dapat ditulis sebagai:

$$\begin{aligned} & \text{Minimalkan / Maksimalkan} && c^T x, \\ & \text{Terhadap batasan} && Ax \leq b, \\ & && x \in \mathbb{Z}^n \text{ (atau} \\ & && \text{sebagian variabel} \\ & && \text{bilangan bulat)} \quad \dots\dots\dots (2) \end{aligned}$$

di mana:

- x adalah vektor variabel keputusan bilangan bulat
- c adalah vektor koefisien fungsi objektif
- A adalah matriks batasan
- b adalah vektor batasan
- \mathbb{Z} Himpunan bilangan bulat

Terdapat tiga jenis utama IP. Pertama adalah *Pure Integer Programming*, yang dimana semua variabel harus bilangan bulat; *Mixed-Integer Programming* (MIP): Sebagian variabel bilangan bulat, sebagian lain kontinu; serta *Binary Integer Programming* (0-1 IP): Variabel hanya bernilai 0 atau 1, berguna untuk keputusan biner (misal: memilih atau tidak memilih suatu item).

2.1.5 Mixed-Integer Linear Programming

MILP adalah variasi dari Integer Programming dimana sebagian variabel harus bernilai bilangan bulat dan sebagian lain boleh bernilai kontinu. Selain itu, fungsi objektif serta batasan harus bersifat linier. Bentuk umum masalah MILP adalah sebagai berikut:

$$\text{Minimalkan / Maksimalkan } c^T x + d^T y,$$

$$\text{Terhadap batasan } Ax + By \leq b,$$

$$x_i \in \mathbb{Z} \quad \forall i \in I,$$

$$y_j \in \mathbb{R} \quad \forall j \in J, \quad \dots\dots (3)$$

keterangan:

- x adalah vektor variabel bilangan bulat
- y adalah vektor variabel bilangan kontinu
- I, J adalah himpunan indeks untuk variabel bilangan bulat dan kontinyu

2.1.6 Multi Objective Optimization

Multi-Objective Optimization (MOO) merupakan pendekatan optimasi yang melibatkan beberapa fungsi tujuan yang seringkali saling bertentangan. Salah satu metode paling dasar dan umum digunakan pada MOO adalah *Weighted Sum Approach*. Metode ini mengubah masalah *multi-objective* menjadi *single-objective* dengan memberikan bobot (*weight*) pada setiap fungsi tujuan, kemudian menjumlahkannya secara linear. Secara matematis, jika terdapat sebanyak k fungsi tujuan $f_1(x), f_2(x), \dots, f_k(x)$, maka formulasi pendekatan *weighted sum* adalah:

$$\text{Minimalisir } F(x) = w_1 f_1(x) + w_2 f_2(x) + \dots + w_k f_k(x) \dots\dots (4)$$

Dengan w_i adalah bobot non-negatif yang memenuhi $\sum_{i=1}^k w_i = 1$.

Bobot ini merefleksikan kepentingan relatif dari setiap tujuan; semakin besar w_i , semakin tinggi prioritas tujuan ke- i dalam proses optimisasi.

Keunggulan utama dari weighted sum approach adalah kesederhanaannya. Metode ini mudah diimplementasikan karena hanya memerlukan penyelesaian permasalahan *single-objective optimization*, sehingga teknik-teknik standar seperti *linear programming* atau *nonlinear programming* dapat langsung diaplikasikan. Selain itu, metode ini fleksibel dalam mengeksplorasi *trade-off* antara berbagai tujuan dengan memvariasikan nilai bobot.

Namun, terdapat beberapa keterbatasan yang perlu diperhatikan. Pertama, pemilihan bobot yang tepat seringkali bersifat subjektif dan memerlukan pengetahuan domain yang mendalam. Kedua, metode ini tidak selalu mampu menemukan semua solusi optimal (*Pareto optimal*) terutama untuk masalah non-convex, karena sifatnya yang hanya memberikan satu solusi untuk setiap kombinasi bobot. Ketiga, skala dari masing-masing fungsi tujuan dapat mempengaruhi hasil secara signifikan, sehingga normalisasi seringkali diperlukan sebelum pembobotan.

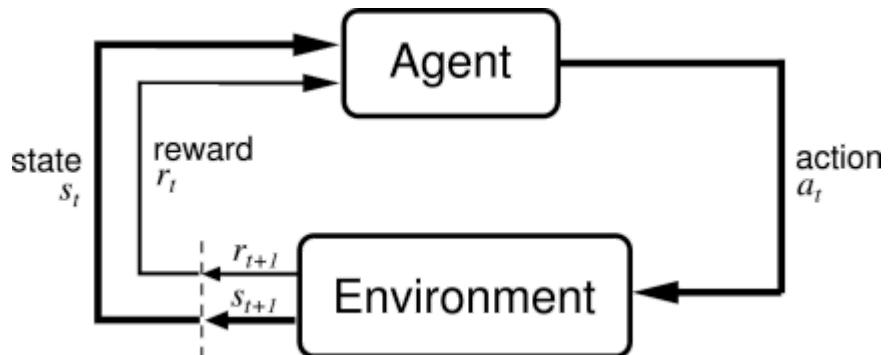
Xin-She Yang (2008) dalam bukunya memberikan contoh aplikasi *weighted sum approach* dalam desain teknik, di mana tujuan untuk meminimalkan biaya dan memaksimalkan kekuatan struktur diberi bobot berbeda. Dalam kasus ini, solusi yang diperoleh akan bergantung pada preferensi apakah lebih mementingkan aspek ekonomi atau performa teknis.

2.1.7 Reinforcement Learning (RL)

Kendala dari metode MILP yang dibahas pada tinjauan pustaka sejauh ini adalah bahwa pendekatan optimasi matematika tidak

menghasilkan suatu model yang dapat digunakan setelah pelatihan. Bagian ini akan kemudian membahas alternatif untuk permasalahan optimasi dengan ruang aksi diskrit berbasis model.

Reinforcement Learning (RL) adalah salah satu cabang dari *Machine Learning* yang berfokus pada bagaimana membuat sebuah agen pintar dapat belajar dari apa yang agen itu lakukan ada sebuah *Environment* (Lingkungan). *Reinforcement Learning* memungkinkan agen perangkat lunak untuk belajar dari interaksi dengan lingkungan dan terus meningkatkan kebijakan pengambilan keputusan berdasarkan umpan balik yang diterima. Dalam prosesnya, agen berinteraksi dengan lingkungan dan menerima informasi yang merepresentasikan status lingkungannya berupa *state* dan sebuah imbalan (*reward*) atau hukuman berupa nilai negatif. Kedua informasi ini kemudian akan diterima kembali oleh agen dan dipelajari untuk menentukan aksi selanjutnya (Sutton & Barto, 2018).



Gambar 2. Diagram Umum *Reinforcement Learning*

RL mengasumsikan bahwa permasalahan dapat dimodelkan sebagai *Markov Decision Process* (MDP). MDP menyediakan kerangka matematis formal untuk memodelkan pengambilan keputusan dalam situasi di mana hasil sebagian ditentukan secara acak dan sebagian di bawah kendali pembuat keputusan. Sifat Markov dari MDP mengasumsikan bahwa informasi dalam keadaan saat ini dan tindakan yang diambil adalah mutlak semua yang diperlukan untuk menentukan

keadaan berikutnya dan hadiah yang akan diterima (Sutton & Barto, 2018).

MDP didefinisikan oleh himpunan keadaan dan tindakan serta dinamika lingkungan satu langkah. Diberikan keadaan (s) dan tindakan tertentu (a), probabilitas setiap pasangan keadaan berikutnya (s') dan imbalan (r), dinotasikan sebagai rumus berikut:

$$Prob(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a) \dots \quad (5)$$

Tujuan perencanaan dalam MDP adalah untuk menemukan *policy* $\pi : S \rightarrow A$, sebuah pemetaan dari keadaan ke tindakan, yang memaksimalkan hadiah diskon yang diharapkan di masa depan ketika agen memilih tindakan sesuai dengan π di lingkungan. Kebijakan yang memaksimalkan hadiah diskon yang diharapkan di masa depan disebut kebijakan optimal dan dilambangkan dengan π^* .

Konsep kunci terkait MDP adalah fungsi Q, $Q\pi : S \times A \rightarrow R$, yang mendefinisikan hadiah diskon yang diharapkan di masa depan untuk melakukan tindakan a dalam keadaan s dan kemudian mengikuti kebijakan π setelahnya. Menurut persamaan Bellman, fungsi Q untuk kebijakan optimal (dilambangkan Q^*) dapat diekspresikan secara rekursif sebagai:

$$Q^*(s, a) = \sum_{s' \in S} T(s, a, s') [R(s, a, s') + \gamma \max_a Q^*(s', a)] \dots \quad (6)$$

di mana $0 \leq \gamma \leq 1$ adalah faktor diskon yang menentukan seberapa berharganya hadiah jangka pendek dibandingkan dengan hadiah jangka panjang. Diberikan Q^* , kebijakan optimal, π^* , dapat dipulihkan dengan mudah dengan memilih tindakan secara serakah (*greedy*) pada keadaan saat ini dengan nilai Q tertinggi: $\pi^*(s) = \arg \max_a Q^*(s, a)$. Properti ini telah mengarah pada berbagai algoritma pembelajaran yang berusaha untuk langsung memperkirakan Q^* , dan memulihkan kebijakan optimal

darinya. Salah satu yang sangat penting adalah Q-Learning (Watkins, 1989).

Algoritma RL seperti *Q-learning*, *Deep Q-Network (DQN)*, dan *Proximal Policy Optimization (PPO)* telah berhasil diterapkan dalam berbagai skenario pengambilan keputusan, termasuk robotika, permainan, dan optimasi sistem (Mnih et al., 2015; Schulman et al., 2017). Dalam konteks penugasan tugas, RL dapat digunakan untuk mempelajari kebijakan penugasan yang optimal berdasarkan data historis dan pengalaman, serta beradaptasi dengan perubahan lingkungan secara langsung.

2.1.8 *Q-Learning*

Q-Learning adalah algoritma *Reinforcement Learning* tanpa model yang bertujuan untuk menemukan kebijakan optimal dengan mempelajari nilai aksi (action-value) untuk setiap pasangan state-aksi. Dalam Q-Learning, nilai aksi atau Q-value, dinyatakan sebagai $Q(s,a)$ merepresentasikan nilai reward total yang dapat diekspektasi sebuah agen semisal pada state s , agen tersebut mengambil aksi a , dan mengikuti keputusan berdasarkan policy sekarang. Nilai Q adalah tolak ukur kualitas aksi dari state tertentu (Watkins, 1992).

Q-Learning memperbarui Q-value menggunakan persamaan Bellman secara iteratif. Setelah mengambil aksi a_t di state s_t , menerima reward r_{t+1} , dan bertransisi ke state s_{t+1} , Q-value diperbarui dengan cara berikut.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max Q(s_{t+1}, a) - Q(s_t, a_t)]. \quad (7)$$

di mana:

- α adalah learning rate ukuran langkah ($0 \leq \alpha \leq 1$)
- γ adalah faktor diskon ($0 \leq \gamma \leq 1$)

- $\max Q(s_{t+1}, a)$ Nilai Q maksimum untuk state berikutnya, mencerminkan aksi terbaik.

Pembaruan ini meminimalkan kesalahan antara Q-value saat ini dan target $r_{t+1} + \gamma \max Q(s_{t+1}, a)$, yang memungkinkan konvergensi ke nilai Q-value optimal pada pembelajaran (Watkins, 1992).

Metode Q-Learning efektif untuk lingkungan dengan ruang state dan aksi yang kecil, karena Q-value disimpan dalam tabel Q. Namun, untuk lingkungan dengan ruang state-aksi besar, seperti dalam konteks penelitian ini dengan 109 tugas dan 300 karyawan, tabel Q menjadi kurang praktis karena memerlukan memori yang sangat besar dan waktu konvergensi lama. Maka dari itu, akan dieksplorasi pada bagian selanjutnya pengembangan *Q-Learning* dengan *Neural Network* yang menggunakan konsep mirip dan lebih memungkinkan untuk diimplementasi untuk ruang state dan aksi yang lebih besar.

2.1.9 Deep *Q-Network* (DQN)

Deep Q-Network (DQN) merupakan salah satu algoritma *Reinforcement Learning* (RL) yang menggabungkan *Q-Learning* dengan jaringan saraf tiruan untuk menangani masalah pengambilan keputusan dalam ruang keadaan yang besar. Dalam konteks optimasi penugasan tugas (*task assignment*), DQN dapat mempelajari kebijakan penugasan yang optimal dengan mempertimbangkan berbagai objektif secara bersamaan. Dalam DQN, Q-value ($Q(s, a; \theta)$) diprediksi oleh *policy network* dengan bobot θ , yang menerima state sebagai input dan menghasilkan Q-value untuk semua aksi yang mungkin. DQN menggunakan dua *neural network*. Network pertama adalah *Policy Network*, yang bertugas untuk memprediksi Q-Value untuk memilih aksi dan diperbarui setiap langkah pelatihan dengan *experience replay*. Jaringan kedua adalah *Target Network*, yang dengan bobot θ^- ,

menghasilkan Q-value untuk state selanjutnya dan diperbarui secara periodik, misal setiap 1000 langkah.

Setelah mengambil aksi a_t di state s_t , menerima reward r_t dan bertransisi ke state s_{t+1} dan memperoleh parameter episode selesai d_t , target Q-value dihitung sebagai berikut.

$$\text{Target Q-Value} = r_t + (1 - d_t) \cdot \gamma \cdot \max_{a'}(s_{t+1}, a'; \theta^-) \dots \dots \dots (8)$$

Pembaruan ini dilakukan menggunakan stochastic gradient descent (SGD) untuk meminimalkan *loss function* berikut:

$$L(\theta) = E[(r + \gamma \max Q(s', a', \theta^-) - Q(s, a; \theta))^2] \dots \dots \dots (9)$$

Bobot θ diperbarui dengan gradient descent untuk meminimalisir nilai loss, yang akan meningkatkan akurasi prediksi Q-value. DQN juga mengimplementasi *experience replay* dan *epsilon-greedy exploration* untuk meningkatkan stabilitas. *Experience replay* menyimpan transisi $(s_t, a_t, r_t, s_{t+1}, d_t)$ dalam memori dan mengambil sampel acak untuk pelatihan *policy* dan *target network*. Disini, s_t digunakan sebagai input untuk *policy network*, sedangkan s_{t+1} akan digunakan sebagai input untuk *target network*. Nilai Q-value yang diberikan kedua network tersebut adalah “label” untuk pelatihan *network* seperti metode *supervised machine learning*. Epsilon-greedy Exploration memilih aksi acak dengan probabilitas ε , yang menurun seiring waktu untuk menyeimbangkan eksplorasi dan eksloitasi pengalaman.

Dalam lingkungan penugasan tugas, DQN efektif untuk menangani ruang aksi yang lebih besar dan state lebih kompleks. Dapat dilihat bentuk umum DQN pada pseudocode berikut yang dikutip dari Mnih *et al.* (2015).

Tabel 1. Pseudocode DQN (sumber: Mnih, 2013)

Algoritma <i>deep Q-Learning</i> dengan <i>experience replay</i>
Inisialisasi replay memory D sebesar kapasitas N
Inisialisasi fungsi <i>action-value</i> Q dengan bobot random θ
Inisialisasi target fungsi <i>action-value</i> \hat{Q} dengan bobot $\theta^- = \theta$
FOR episode = 1, M DO
Inisialisasi urutan $s_1 = \{x_1\}$ & urutan hasil pemrosesan $\Phi_1 = \Phi(s_1)$
FOR episode t = 1, T DO
Dengan probabilitas ϵ pilih aksi random a_t
Bila tidak, pilih $a_t = \operatorname{argmax}_a Q(\phi(s_t), a; \theta)$
Eksekusi aksi a_t pada emulator & observasi reward r_t & image x_{t+1}
Tentukan $s_{t+1} = s_t, a_t, r_t, x_{t+1}$ dan proses $\Phi_{t+1} = \Phi(s_{t+1})$
Setor transisi $(\Phi_t, a_t, r_t, \Phi_{t+1})$ pada D
Sampel mini batch random transisi $(\Phi_t, a_t, r_t, \Phi_{t+1})$ dari D
Tentukan $y_j = r_j$ semisal episode selesai pada step j+1, bila tidak
Tentukan $y_j = r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta^-)$
Lakukan langkah gradient descent pada $(y_j - Q(\phi_j, a_j; \theta))^2$
yang mengikuti parameter network θ
Setiap langkah sebanyak C, tentukan ulang $\hat{Q} = Q$
END FOR
END FOR

2.1.10 Euclidean Distance

Euclidean distance adalah metode yang umum digunakan untuk mengukur kesamaan atau perbedaan antara dua titik dalam ruang multidimensi. Rumus untuk menghitung *Euclidean distance* antara dua titik dalam ruang berdimensi n adalah sebagai berikut:

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2} \quad \dots \dots \dots \quad (10)$$

Keterangan:

n = jumlah komponen pada vektor

y_i = elemen ke-i pada vektor y

x_i = elemen ke-i pada vektor x

Rumus ini menghitung jarak garis lurus antara dua titik. Namun, ketika menerapkan *Euclidean distance* pada masalah tertentu, seperti penugasan tugas kepada karyawan berdasarkan metrik keterampilan, hasilnya tidak sesuai dengan konteks yang ditentukan pada permasalahan ini. Karyawan ideal adalah karyawan yang memiliki kemampuan sama persis dengan kriteria tugas. Karyawan yang berada dibawah kualifikasi dan melebihi kualifikasi sama-sama tidak cocok. Untuk perbedaan poin kualifikasi yang sama, tingkat kecocokan karyawan yang dibawah kualifikasi memiliki angka ketidakcocokan (*similarity score*) lebih besar dibanding yang melebihi kualifikasi untuk perbedaan.

Terdapat tiga jenis karyawan - karyawan dibawah kualifikasi, sempurna, dan diatas kualifikasi. Dapat diambil contoh sebagai berikut:

$employee_{underqualified}$	$= [2,2,2]$
$employee_{perfect}$	$= [3,3,3]$
$employee_{overqualified}$	$= [4,4,4]$
$task$	$= [3,3,3]$

Gambar 3. Contoh Vektor Kualifikasi Karyawan dan Tugas

Rumus *Euclidean Distance* dengan cocok dapat menemukan karyawan yang paling sempurna, tetapi gagal membedakan karyawan yang memiliki perbedaan poin 1 karena di bawah kualifikasi atau diatas kualifikasi.

$$\begin{aligned}\Delta employee_{underqualified} &= \sqrt{(2 - 3)^2 + (2 - 3)^2 + (2 - 3)^2} \\ &= \sqrt{1 + 1 + 1} = \sqrt{3} \approx 1.73\end{aligned}$$

$$\begin{aligned}\Delta employee_{perfect} &= \sqrt{(3 - 3)^2 + (3 - 3)^2 + (3 - 3)^2} \\ &= \sqrt{0 + 0 + 0} = \sqrt{0} \approx 0\end{aligned}$$

$$\begin{aligned}\Delta employee_{overqualified} &= \sqrt{(4 - 3)^2 + (4 - 3)^2 + (4 - 3)^2} \\ &= \sqrt{1 + 1 + 1} = \sqrt{3} \approx 1.73\end{aligned}$$

Gambar 4. Pembuktian Perhitungan *Euclidean Distance*

Metode *Euclidean distance* memberikan skor jarak yang sama untuk karyawan yang dibawah kriteria dan karyawan yang di atas kriteria. Secara matematis, hasil yang didapat tidak sesuai secara konteks. Hal ini terjadi karena *Euclidean distance* mengkuadratkan perbedaan antara

dimensi yang bersesuaian, yang dapat mengabaikan relevansi atau konteks sebenarnya dari masalah tersebut.

Dalam konteks manajemen proyek, apabila hanya diberikan pilihan antara karyawan yang diatas kualifikasi atau dibawah kualifikasi, keputusan yang lebih tepat adalah untuk memberikan tugas tersebut kepada karyawan yang diatas kualifikasi. Namun karyawan yang memiliki kemampuan sama persis sesuai kriteria tugas akan menjadi pilihan ideal dalam konteks ini. Maka dari itu dibutuhkan rumus yang dapat membedakan ketiga karyawan tersebut yang dimana karyawan dengan kemampuan sama persis dengan kriteria tugas adalah karyawan paling cocok, dilanjutkan dengan karyawan yang diatas kualifikasi, lalu karyawan yang dibawah kualifikasi.

2.1.11 Fungsi *Weight* pada *Weighted Euclidean Distance*

Tantangan utama rumus *Euclidean Distance* adalah bagaimana membedakan antara karyawan yang *overqualified* (memiliki keterampilan melebihi yang dibutuhkan) dan *underqualified* (memiliki keterampilan kurang dari yang dibutuhkan), terutama ketika perbedaan absolut antara keterampilan karyawan dan persyaratan tugas adalah sama. *Euclidean Distance* standar memperlakukan semua dimensi keterampilan secara setara, sehingga tidak dapat menangkap nuansa perbedaan kualifikasi ini. Oleh karena itu, diperlukan suatu fungsi bobot yang dapat memberikan penalti terhadap kelebihan kualifikasi, namun tetap memberikan bobot yang wajar bagi karyawan yang kurang memenuhi syarat atau memiliki kecocokan sempurna dengan tugas.

Untuk mengatasi masalah ini, penulis mengadopsi konsep fungsi keanggotaan fuzzy. Himpunan fuzzy, berbeda dengan himpunan klasik, memungkinkan adanya derajat keanggotaan parsial, di mana suatu elemen suatu himpunan adalah 0 (tidak termasuk sama sekali) hingga 1 (termasuk sepenuhnya) (Klir & Yuan, 1995). Fungsi keanggotaan fuzzy memetakan nilai input (dalam hal ini, tingkat keterampilan karyawan) ke nilai

keanggotaan antara 0 dan 1, yang mencerminkan sejauh mana elemen tersebut termasuk dalam himpunan fuzzy. Fungsi keanggotaan fuzzy yang sederhana dapat dilihat sebagai berikut

$$membership(x, ideal) = \frac{1}{(1 + distance(x, ideal))} \dots \dots \dots (11)$$

Keterangan:

x = nilai input (keterampilan karyawan)

ideal = nilai Ideal (keterampilan yang dibutuhkan)

`distance(x,ideal)` = jarak antara x dan Ideal

Fungsi dasar keanggotaan fuzzy perlu dimodifikasi supaya sesuai dengan konteks penugasan tugas.

1. Jarak: menggunakan selisih $e - t$ untuk mengukur jarak antara keterampilan karyawan dan persyaratan tugas. Namun, karena hanya kelebihan kualifikasi yang ingin diberikan penalti, maka digunakan $\max(e-t)$ untuk memastikan jaraknya non-negatif.
 2. Kontrol Penalti: Parameter α akan mengontrol tingkat penalti bagi kelebihan kualifikasi. Jarak dikalikan dengan α untuk mengatur dampaknya pada bobot. Dalam penelitian ini, dipilih parameter 0.5 sebagai nilai awal yang memberikan keseimbangan yang baik antara menghukum kelebihan kualifikasi dan tetap mempertimbangkan keterampilan yang relevan.
 3. Normalisasi: Mempertahankan struktur $1 / (1 + \dots)$ supaya bobot tetap berada di antara 0 dan 1.

Hasil modifikasi yang disesuaikan dengan konteks permasalahan akan memberikan definisi sebagai berikut:

$$weight = \frac{1}{1 + \alpha (\max(0, (e - t)))} \quad \dots \dots \dots (12)$$

Keterangan:

- n = Jumlah komponen pada vektor
 e_i = Elemen ke- i pada vektor karyawan (*employee*)
 t_i = Elemen ke- i pada vektor tugas (*task*)
 α = parameter yang mengontrol kepentingan bobot
overqualification

2.1.12 Weighted Euclidean Distance

Weighted Euclidean distance adalah modifikasi dari rumus *Euclidean distance* standar yang menggabungkan bobot atau faktor kepentingan untuk setiap dimensi atau fitur. Rumus *Euclidean* standar tidak cocok untuk masalah penugasan tugas ini karena tidak membedakan secara efektif antara karyawan yang diatas kualifikasi dan yang dibawah kualifikasi. *Weighted Euclidean distance* mengatasi masalah ini dengan memungkinkan setiap dimensi diukur dengan bobot yang sesuai, yang mencerminkan kepentingan relatif dari dimensi tersebut dalam perhitungan jarak. Modifikasi ini memastikan bahwa ukuran jarak memperhitungkan berbagai tingkat kualifikasi secara lebih akurat, sehingga meningkatkan proses penugasan tugas.

Dalam studi ini, penulis mengadopsi fungsi bobot yang diperberat untuk memberikan bobot kepada karyawan berdasarkan tingkat keterampilan mereka relatif terhadap keterampilan tugas yang dibutuhkan. Pendekatan ini terinspirasi oleh prinsip-prinsip yang diuraikan dalam (Taha, 2017). Nilai alpha, yang merupakan parameter penyetelan yang dapat disesuaikan oleh pengguna, akan digunakan untuk mengoptimalkan kinerja dengan menyempurnakan bobot ini.

Berikut adalah rumus untuk weighted Euclidean distance, yang secara eksplisit menunjukkan bagaimana bobot diterapkan pada setiap dimensi:

$$d(e, t) = \sqrt{\sum_{i=0}^n w(e_i - t_i)^2} \quad \dots \dots \dots \quad (13)$$

Keterangan:

n = Jumlah komponen pada vektor

e_i = Elemen ke- i pada vektor karyawan (*employee*)

t_i = Elemen ke- i pada vektor tugas (*task*)

w = Bobot komponen

Tantangan metode rumus *Euclidean Distance* yang tidak dapat membedakan karyawan yang dibawah kualifikasi atau diatas dapat diatas menggunakan metode *Weighted Euclidean Distance*. Berikut adalah pembuktian matematis untuk permasalahan yang sama.

$\omega_{employee_{underqualified}}$

$$\omega_1 = \frac{1}{1 + 0.5 \times (\max(0, 2-3))} = \frac{1}{1 + 0.5 \times (0)} = \frac{1}{1} = 1$$

$$\omega_2 = \frac{1}{1 + 0.5 \times (\max(0, 2-3))} = \frac{1}{1 + 0.5 \times (0)} = \frac{1}{1} = 1$$

$$\omega_3 = \frac{1}{1 + 0.5 \times (\max(0, 2-3))} = \frac{1}{1 + 0.5 \times (0)} = \frac{1}{1} = 1$$

$\omega_{employee_{perfect}}$

$$\omega_1 = \frac{1}{1 + 0.5 \times (\max(0, 3-3))} = \frac{1}{1 + 0.5 \times (0)} = \frac{1}{1} = 1$$

$$\omega_2 = \frac{1}{1 + 0.5 \times (\max(0, 3-3))} = \frac{1}{1 + 0.5 \times (0)} = \frac{1}{1} = 1$$

$$\omega_3 = \frac{1}{1 + 0.5 \times (\max(0, 3-3))} = \frac{1}{1 + 0.5 \times (0)} = \frac{1}{1} = 1$$

$\omega_{employee_{overqualified}}$

$$\omega_1 = \frac{1}{1 + 0.5 \times (\max(0, 4-3))} = \frac{1}{1 + 0.5 \times (1)} = \frac{1}{1.5} = 0.66$$

$$\omega_2 = \frac{1}{1+0.5 \times (\max(0, 4-3))} = \frac{1}{1+0.5 \times (1)} = \frac{1}{1.5} = 0.66$$

$$\omega_3 = \frac{1}{1+0.5 \times (\max(0, 4-3))} = \frac{1}{1+0.5 \times (1)} = \frac{1}{1.5} = 0.66$$

Gambar 5. Perhitungan *Weight* pada *Weighted Euclidean Distance*

Ketiga *Weight* untuk masing-masing employee kebetulan sama untuk kasus ini. Berikutnya, nilai masing-masing weight akan digunakan pada rumus *Weighted Euclidean Distance*. Berikut adalah pembuktian perhitungan untuk masing-masing karyawan terhadap tugas yang sama:

$$\Delta_{employee_{underqualified}} = \sqrt{1 * (2 - 3)^2 + 1 * (2 - 3)^2 + 1 * (2 - 3)^2} = \sqrt{1 + 1 + 1} = \sqrt{3} \approx 1.73$$

$$\Delta_{employee_{perfect}} = \sqrt{1 * (3 - 3)^2 + 1 * (3 - 3)^2 + 1 * (3 - 3)^2} = \sqrt{0 + 0 + 0} = \sqrt{0} \approx 0$$

$$\Delta_{employee_{overqualified}} = \sqrt{0.66 * (4 - 3)^2 + 0.66 * (4 - 3)^2 + 0.66 * (4 - 3)^2} = \sqrt{0.66 + 0.66 + 0.66} \approx 1.41$$

Gambar 6. Pembuktian Perhitungan *Weighted Euclidean Distance*

Karyawan yang memiliki kemampuan dibawah kualifikasi ([2, 2, 2]) dan diatas kualifikasi ([4, 4, 4]) sama-sama memiliki perbedaan sebanyak satu poin terhadap vektor tugas [3, 3, 3]. Pembuktian perhitungan pada gambar 6 menunjukkan bahwa dengan perbedaan yang sama, karyawan yang diatas kualifikasi lebih baik daripada karyawan yang dibawah kualifikasi. Namun tidak lebih baik daripada karyawan yang memiliki kualifikasi yang sempurna.

2.1.13 Standar Deviasi

Standar deviasi merupakan ukuran dispersi statistik yang mengkuantifikasi seberapa jauh titik data tersebar dari nilai rata-ratanya. Dalam konteks optimasi, standar deviasi sering digunakan sebagai metrik untuk mengevaluasi stabilitas solusi, ketidakpastian parameter, atau variasi performa algoritma. Secara matematis, standar deviasi (σ) dari suatu himpunan data x_1, x_2, \dots, x_n didefinisikan sebagai akar kuadrat dari varians:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} \quad \dots \dots \dots \quad (14)$$

Dimana \bar{x} adalah rata-rata sampel. Dalam optimisasi stokastik atau algoritma metaheuristik, standar deviasi digunakan untuk menganalisis konsistensi solusi yang dihasilkan dari berbagai run, terutama ketika algoritma memiliki komponen acak (Xin-She Yang, 2008).

2.1.14 Perbandingan Metode Pengukuran Jarak Lainnya

Picture	Method	Features	Disadvantages	Formula
	Minkowski Distance	Measures the distance between two points in n-dimentional space, where r determine the metric used	Better to choose manhattan or euclidian only	$D(A, B) = (\sum_{i=1}^n A_i - B_i ^r)^{\frac{1}{r}}$
	Chebyshev Distance	Measures the maximum difference between corresponding components of two vectors	Not applicable	$D(A, B) = \max(A_i - B_i)$
	Jaccard Similarity	Measures the similarity between two sets by comparing their intersection and union	Ignore magnitudes of sets	$J(A, B) = \frac{ A \cap B }{ A \cup B }$
	Pearson Correlation	Measures the linear correlation between two variables in a dataset	Not useful for high dimentional data	$r = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum (X_i - \bar{X})^2 \sum (Y_i - \bar{Y})^2}}$
	Sørensen-Dice Index	Measures the similarity between two sets	Ignore magnitudes of sets	$DSC(A, B) = \frac{2 \times A \cap B }{ A + B }$
	Manhattan Distance	Measures the distance between two points on a grid network, where movement is limited	Not useful for high dimentional data	$D(A, B) = \sum_{i=1}^n A_i - B_i $

Gambar 7. Perbandingan metode penghitungan jarak kecocokan vektor

Metode-metode yang dipertimbangkan meliputi Minkowski Distance, Chebyshev Distance, Jaccard Similarity, Pearson Correlation, Sørensen-Dice Index, dan Manhattan Distance.

1. *Minkowski Distance*: Metode ini mengukur jarak antara dua titik dalam ruang berdimensi n, dengan parameter r yang menentukan metrik yang digunakan. Namun, ditemukan bahwa menggunakan hanya *Manhattan* atau *Euclidean distance* lebih disukai untuk tujuan penulis, karena *Minkowski Distance* tidak menawarkan keuntungan signifikan dalam konteks penelitian.
2. *Chebyshev Distance*: Metode ini mengukur perbedaan maksimum antara komponen yang bersesuaian dari dua vektor. Sayangnya, ini tidak berlaku untuk dataset kami karena tidak dapat secara efektif menangkap nuansa keterampilan karyawan dan persyaratan tugas.
3. *Jaccard Similarity*: Metode ini mengukur kesamaan antara dua set dengan membandingkan irisan dan gabungan mereka. Meskipun berguna untuk jenis perbandingan tertentu, ini mengabaikan besaran set, menjadikannya tidak cocok untuk model optimisasi kami yang membutuhkan pengukuran tingkat keterampilan yang tepat.
4. *Pearson Correlation*: Metode ini mengukur korelasi linier antara dua variabel dalam dataset. Ini dianggap tidak berguna untuk data berdimensi tinggi, karena dataset kami melibatkan banyak dimensi dan hubungan kompleks antara keterampilan karyawan dan persyaratan tugas.
5. *Sørensen-Dice Index*: Metode ini mengukur kesamaan antara dua set, mirip dengan *Jaccard Similarity*. Ini juga mengabaikan besaran set, sehingga gagal memberikan presisi yang diperlukan untuk optimisasi penugasan tugas kami.
6. *Manhattan Distance*: Metode ini mengukur jarak antara dua titik pada jaringan grid di mana gerakan dibatasi. Meskipun dapat berguna dalam skenario tertentu, ini tidak cocok untuk data berdimensi tinggi kami.

2.2 Tinjauan Empiris

2.2.1 A MILP model for the Teacher Assignment Problem considering teachers' preferences

Penelitian oleh Domenech dan Lusa (2016) mengembangkan pendekatan *Mixed Integer Linear Programming* (MILP) untuk menangani masalah penugasan guru ke mata pelajaran dengan mempertimbangkan preferensi individu mereka. Masalah ini merupakan bagian dari *University Timetabling Problem*, yang dikenal sebagai permasalahan kompleks dan termasuk dalam kategori *NP-hard* karena melibatkan banyak variabel dan kendala. Penelitian ini merancang model MILP dengan tujuan utama untuk menyeimbangkan beban mengajar guru (*teaching load*), yang menjadi kriteria optimasi pertama, sambil memaksimalkan kepuasan preferensi guru terhadap mata pelajaran sesuai dengan kategori mereka sebagai kriteria optimasi kedua. Untuk mencapai tujuan tersebut, model ini mempertimbangkan berbagai parameter seperti jumlah jam mengajar, kualifikasi guru, dan preferensi spesifik terhadap mata pelajaran tertentu.

Eksperimen komputasi dilakukan dengan menggunakan data berbasis pola nyata, melibatkan skenario hingga 40 guru dan 120 mata pelajaran. Hasilnya menunjukkan bahwa model MILP yang dikembangkan mampu menghasilkan solusi optimal atau mendekati optimal dalam waktu komputasi yang relatif wajar, bahkan untuk kasus dengan skala yang cukup besar. Selain itu, penelitian ini juga menganalisis sensitivitas model terhadap perubahan parameter, seperti variasi jumlah guru atau mata pelajaran. Temuan ini menunjukkan bahwa pendekatan MILP dapat secara efektif menangani masalah penugasan dengan mempertimbangkan preferensi, meskipun membutuhkan waktu komputasi yang meningkat seiring dengan bertambahnya kompleksitas masalah.

Relevansi penelitian ini terhadap penelitian yang sedang dilakukan terletak pada pendekatan MILP yang dapat diadaptasi untuk masalah

penugasan tugas kepada karyawan dengan mempertimbangkan preferensi mereka serta keseimbangan beban kerja. Meskipun konteksnya berbeda - yaitu dunia pendidikan, prinsip dasar model ini, seperti formulasi matematis untuk optimasi multi-kriteria, dapat dijadikan acuan dalam merancang solusi untuk penugasan tugas karyawan. Penelitian ini juga memberikan wawasan tentang pentingnya mempertimbangkan *trade-off* antara keseimbangan beban kerja dan kepuasan preferensi, yang menjadi fokus utama dalam penelitian yang sedang dilakukan.

2.2.2 Reinforcement Learning and Mixed-Integer Programming for Power Plant Scheduling in Low Carbon Systems: Comparison and Hybridisation

Penelitian oleh O’Malley (2023) mengeksplorasi perbandingan serta potensi hibridisasi antara *Reinforcement Learning* (RL) dan *Mixed Integer Linear Programming* (MILP) dalam konteks penjadwalan pembangkit listrik untuk sistem rendah karbon. Penelitian ini menyoroti bahwa MILP, sebagai pendekatan deterministik, sering kali menghadapi tantangan dalam situasi yang sangat dinamis dan kompleks, seperti penjadwalan pembangkit listrik dengan permintaan energi yang fluktuatif dan ketidakpastian sumber energi terbarukan. Sebaliknya, RL menawarkan fleksibilitas adaptif dengan kemampuan untuk belajar dari lingkungan secara iteratif, sehingga dapat menjadi alternatif yang potensial.

Penelitian ini menguji kedua pendekatan tersebut secara terpisah sebelum mengevaluasi pendekatan hibrid yang menggabungkan keunggulan MILP dan RL. Hasilnya menunjukkan bahwa MILP mampu memberikan solusi optimal dalam skenario yang lebih sederhana dengan kendala yang terdefinisi dengan baik, tetapi performanya menurun ketika menghadapi dinamika tinggi dan ketidakpastian. Di sisi lain, RL menunjukkan kemampuan yang lebih baik dalam beradaptasi terhadap perubahan kondisi secara langsung, meskipun memerlukan waktu

pelatihan yang lebih lama untuk mencapai performa optimal. Pendekatan hibrid yang diusulkan dalam penelitian ini memanfaatkan MILP untuk menghasilkan solusi awal yang baik, yang kemudian disempurnakan oleh RL untuk meningkatkan efisiensi dan adaptabilitas. Eksperimen dilakukan dengan simulasi berbasis data nyata dari sistem pembangkit listrik rendah karbon, menunjukkan bahwa pendekatan hibrid dapat meningkatkan efisiensi penjadwalan hingga 15% dibandingkan penggunaan MILP atau RL secara mandiri.

Relevansi penelitian ini sangat signifikan bagi penelitian yang sedang dilakukan karena juga membandingkan pendekatan MILP dan RL dalam konteks penugasan tugas karyawan. Penelitian ini memberikan wawasan tentang keunggulan RL dalam menangani dinamika dan ketidakpastian, yang mungkin relevan dalam penugasan tugas karyawan di lingkungan kerja yang berubah-ubah. Selain itu, pendekatan hibrid yang diusulkan dapat menjadi inspirasi untuk mengintegrasikan MILP dan RL dalam penelitian yang sedang dilakukan, terutama untuk meningkatkan efisiensi dan fleksibilitas solusi yang dihasilkan.

2.2.3 A Deep Reinforcement Learning Approach for Chemical Production Scheduling

Penelitian oleh Hubbs *et al.* (2022) membahas penerapan Deep Reinforcement Learning (DRL) untuk mengoptimalkan penjadwalan produksi dalam industri kimia. Penelitian ini menekankan bahwa lingkungan penjadwalan di industri kimia sering kali bersifat dinamis dan kompleks, dengan faktor-faktor seperti perubahan permintaan, gangguan mesin, dan variasi waktu produksi yang mempengaruhi efisiensi. Secara tradisional, masalah ini diatasi dengan pendekatan MILP, tetapi pendekatan tersebut seringkali kesulitan menangani ketidakpastian dan dinamika secara langsung.

Penelitian ini mengusulkan penggunaan DRL, yang mengintegrasikan *Reinforcement Learning* dengan *Deep Neural Networks*, untuk mengatasi tantangan tersebut. DRL memungkinkan agen untuk belajar kebijakan optimal melalui interaksi langsung dengan lingkungan simulasi, sehingga dapat menghasilkan jadwal produksi yang adaptif terhadap perubahan kondisi. Eksperimen dilakukan dengan membandingkan performa DRL terhadap pendekatan MILP pada kasus penjadwalan produksi dengan hingga 50 mesin dan 200 pesanan. Hasilnya menunjukkan bahwa DRL mampu menghasilkan jadwal yang lebih efisien dalam hal waktu penyelesaian dan pemanfaatan sumber daya, dengan peningkatan performa hingga 10% dibandingkan MILP dalam skenario dinamis. Selain itu, DRL juga menunjukkan kemampuan untuk menangani gangguan secara langsung tanpa memerlukan perhitungan ulang dari awal, yang merupakan kelemahan utama MILP.

Relevansi penelitian ini terhadap penelitian yang sedang dilakukan terletak pada konfirmasinya bahwa DRL dapat menjadi pendekatan yang efektif untuk masalah penjadwalan atau penugasan yang dinamis, seperti penugasan tugas karyawan. Meskipun fokusnya pada industri kimia, prinsip dasar DRL dalam penelitian ini - seperti kemampuan adaptasi dan pembelajaran dari lingkungan, dapat diterapkan pada konteks penugasan tugas karyawan yang melibatkan preferensi dan beban kerja. Penelitian ini juga memperkuat argumen bahwa RL, khususnya DRL, dapat menjadi alternatif yang layak dibandingkan MILP dalam situasi yang membutuhkan fleksibilitas tinggi.

2.2.4 Exploratory Combinatorial Optimization with Reinforcement Learning

Penelitian oleh Barrett et al. (2020) mengeksplorasi penerapan *Reinforcement Learning* (RL), khususnya metode *Deep Q-Network* (DQN), untuk menyelesaikan masalah optimasi kombinatorial seperti *Maximum Cut* pada graf. Berbeda dengan pendekatan tradisional yang

membangun solusi secara bertahap, penelitian ini mengusulkan pendekatan eksploratif di mana agen RL dapat menambah atau menghapus elemen solusi secara iteratif untuk mencari solusi yang lebih baik. Pendekatan ini memungkinkan agen untuk menjelajahi ruang solusi secara lebih fleksibel, sehingga dapat menemukan solusi yang lebih optimal dalam masalah yang kompleks.

Penelitian ini menguji DQN pada berbagai ukuran graf, mulai dari graf kecil dengan 50 simpul hingga graf besar dengan 2000 simpul. Hasilnya menunjukkan bahwa DQN mampu menghasilkan solusi yang kompetitif dibandingkan dengan metode optimasi tradisional seperti *Greedy Randomized Adaptive Search Procedure* (GRASP), dengan keunggulan utama pada kemampuan DQN untuk belajar dari pengalamannya sebelumnya. Selain itu, penelitian ini juga mengeksplorasi kombinasi DQN dengan metode pencarian lain, seperti *local search*, untuk meningkatkan performa lebih lanjut. Temuan ini menunjukkan bahwa pendekatan RL berbasis DQN dapat secara efektif menangani masalah optimasi kombinatorial yang kompleks, bahkan dalam skala besar, dengan waktu komputasi yang kompetitif.

Relevansi penelitian ini terhadap penelitian yang sedang dilakukan adalah konfirmasinya bahwa DQN, sebagai salah satu teknik RL, dapat diterapkan pada masalah optimasi kombinatorial yang memiliki kesamaan dengan penugasan tugas karyawan. Penugasan tugas karyawan dapat dipandang sebagai masalah kombinatorial yang melibatkan banyak variabel, seperti preferensi karyawan dan keseimbangan beban kerja, sehingga pendekatan DQN dapat menjadi alternatif yang potensial untuk diuji. Penelitian ini juga memberikan bukti tambahan bahwa RL dapat menjadi pendekatan yang layak untuk dibandingkan dengan MILP dalam penelitian yang sedang dilakukan, terutama dalam hal fleksibilitas dan kemampuan eksplorasi solusi.

2.2.5 A Comparative Study of Deep Reinforcement Learning Models: DQN vs PPO vs A2C

Penelitian oleh de la Fuente dan Vidal (2024) membandingkan tiga model *Deep Reinforcement Learning* (DRL) - *Deep Q-Networks* (DQN), *Proximal Policy Optimization* (PPO), dan *Advantage Actor-Critic* (A2C) dalam lingkungan permainan BreakOut Atari menggunakan implementasi dari Stable Baselines3. Studi ini mengevaluasi efisiensi pembelajaran, stabilitas, dan optimasi reward melalui eksperimen dengan variasi hyperparameter, yaitu learning rate (1e-5 hingga 5e-3) dan *gamma* discount factor (0.90 dan 0.99), selama pelatihan sebanyak 20 juta frame. Hasilnya menunjukkan bahwa DQN secara konsisten mengungguli PPO dan A2C dalam hal efisiensi pembelajaran, stabilitas, dan reward rata-rata per episode, meskipun memiliki arsitektur yang lebih sederhana.

DQN, yang berbasis *Q-Learning*, memanfaatkan experience replay untuk menyimpan transisi (state, action, reward, next state) dan mengambil sampel acak, mengurangi korelasi data dan meningkatkan generalisasi. Dalam BreakOut, DQN menunjukkan kurva pembelajaran yang lebih halus dan efisiensi penggunaan frame yang lebih tinggi, mencapai reward tinggi dalam waktu pelatihan yang lebih singkat dibandingkan PPO dan A2C. PPO, dengan mekanisme *clipped objective* untuk mencegah pembaruan kebijakan yang terlalu besar, menunjukkan stabilitas yang baik namun membutuhkan episode yang lebih panjang, menandakan pendekatan yang lebih eksploratif. A2C, sebagai metode on-policy, sangat sensitif terhadap learning rate, dengan performa yang bervariasi signifikan dan kurva pembelajaran yang lebih fluktuatif, menunjukkan efisiensi yang lebih rendah. DQN juga lebih tahan terhadap variasi hyperparameter, berkat *experience replay* dan *fixed Q-targets*, sementara PPO dan A2C memerlukan penyetelan yang lebih cermat.

Keunggulan DQN terletak pada kemampuannya mengeksplorasi struktur reward yang jelas di BreakOut, yang selaras dengan pendekatan

value estimation-nya. Berbeda dengan PPO dan A2C, yang mengestimasi kebijakan langsung melalui *policy gradient* dan memerlukan eksplorasi strategi kompleks seperti "tunneling," DQN dapat secara efektif memetakan aksi langsung ke reward. Studi ini menyimpulkan bahwa DQN sangat cocok untuk lingkungan dengan reward langsung dan dinamika sederhana, sementara PPO dan A2C lebih unggul dalam lingkungan kompleks yang membutuhkan eksplorasi mendalam.

Relevansi penelitian ini dengan penelitian yang sedang dilakukan adalah pada potensi DQN untuk menangani lingkungan penugasan tugas karyawan, yang memiliki ruang observasi berdimensi tinggi (matriks penugasan dan kesamaan 500×200 serta fitur global) dan ruang aksi diskret yang besar ($2 \times 500 \times 200$).

2.2.6 A Multiple-Input Neural Network Model for Predicting Cotton Production Quantity: A Case Study

Penelitian oleh Livieris et al. (2020) mengusulkan *Multiple-Input Neural Network* (MNN), sebuah model *neural network* untuk memprediksi hasil panen kapas berdasarkan tiga jenis data: data tanah, data manajemen budidaya, dan data manajemen hasil panen. Model ini memproses ketiga jenis data melalui lapisan input dan *hidden layer* terpisah, kemudian menggabungkan (*concatenate*) outputnya melalui concatenate layer sebagai input untuk lapisan *dense* dan output akhir. Arsitektur ini memungkinkan pemrosesan independen untuk data heterogen.

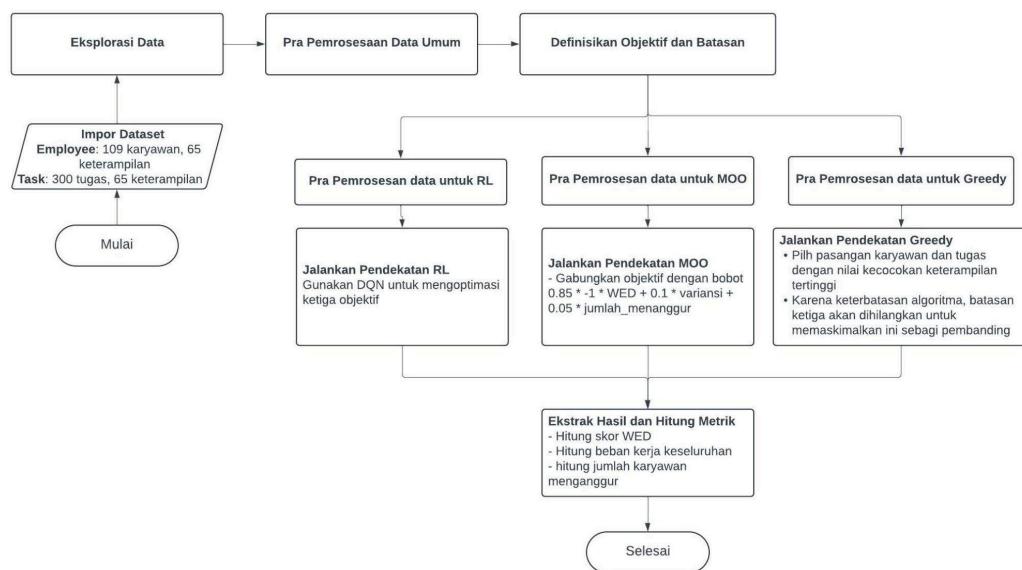
Relevansi penelitian ini dengan penelitian yang sedang dilakukan adalah pada arsitektur MNN yang mencerminkan pendekatan model dalam penelitian ini, di mana tiga state (matriks penugasan, matriks kesamaan, fitur global) diproses secara terpisah melalui transformasi linier, lalu outputnya digabungkan sebagai input jaringan saraf utama. Pendekatan concatenation ini memungkinkan model untuk menangani state berdimensi tinggi (matriks 500×200) dengan efisien, mendukung pembelajaran kebijakan optimal menggunakan DQN.

BAB III

METODE PENELITIAN

3.1 Gambaran Umum

Penelitian ini berfokus pada optimasi penugasan tugas kepada karyawan dengan mempertimbangkan keterampilan dan kompetensi mereka. Penugasan tugas yang efektif sangat penting untuk meningkatkan kinerja perusahaan dan memastikan bahwa setiap karyawan ditempatkan pada peran yang sesuai dengan kemampuan mereka. Gambaran umum pendekatan penelitian ini dapat direpresentasikan pada diagram dibawah ini.



Gambar 8. Diagram Gambaran Umum Metodologi

Metode penelitian dimulai dengan melakukan pra pemrosesan pada data dan mendefinisikan objektif serta batasan menjadi bentuk yang dapat dikomputasikan dengan bentuk matematis. Terdapat tiga pendekatan yang akan digunakan sebagai perbandingan. Pendekatan pertama adalah pendekatan *greedy* yang dimana pencocokan tugas hanya dilakukan berdasarkan nilai WED terbaik. Khusus pendekatan ini akan diberikan kelonggaran batasan bahwa satu karyawan hanya dapat mengambil tugas dari proyek yang sama untuk dapat melihat nilai WED tertinggi yang dapat didapatkan untuk permasalahan ini.

Pendekatan berikutnya adalah pendekatan MOO yang akan melakukan optimasi matematis yang memenuhi semua batasan dan ketiga objektif. Pendekatan RL adalah pendekatan utama dan merupakan topik riset utama pada penelitian ini. Hasil yang didapatkan dari pendekatan *greedy* dan MOO akan menjadi acuan utama untuk performa RL pada tahap evaluasi.

3.2 Objektif

Penugasan tugas dalam penelitian ini bertujuan untuk mengoptimalkan tiga tujuan utama yang mencerminkan prioritas organisasi dalam alokasi sumber daya manusia:

1. Meminimalisir Karyawan yang Tidak Bekerja. Objektif ini memprioritaskan penugasan tugas kepada karyawan yang belum memiliki beban kerja (*idle*), sehingga memaksimalkan pemanfaatan tenaga kerja. Hal ini dilakukan dengan meminimalkan jumlah karyawan yang tidak diberikan tugas, dengan bobot 0,05 dalam fungsi evaluasi, mencerminkan tujuan untuk meningkatkan efisiensi organisasi melalui pemanfaatan sumber daya yang optimal.
2. Memaksimalkan Kesesuaian Keterampilan. Objektif ini berfokus pada pencocokan tugas dengan karyawan berdasarkan keselarasan keterampilan, yang diukur menggunakan skor kesamaan *Weighted Euclidean Distance* (WED). Skor WED merepresentasikan tingkat kecocokan antara vektor keterampilan karyawan dan kebutuhan keterampilan tugas, dengan nilai yang lebih tinggi menunjukkan kesesuaian yang lebih baik. Objektif ini memiliki bobot 0,85 dalam fungsi evaluasi, mencerminkan prioritas utama untuk memastikan efisiensi dan kualitas penyelesaian tugas.
3. Meminimalkan Variansi Beban Kerja. Objektif ini bertujuan untuk mendistribusikan beban kerja secara merata di antara karyawan, sehingga mengurangi ketimpangan dalam alokasi tugas. Beban kerja diukur dalam satuan story points, dan variansi yang lebih kecil menunjukkan distribusi

yang lebih seimbang. Objektif ini diberi bobot 0,1, menunjukkan pentingnya keseimbangan beban kerja sebagai tujuan sekunder.

3.2 Batasan

Penugasan tugas dalam penelitian ini tunduk pada beberapa batasan berikut untuk memastikan kelayakan dan konsistensi solusi:

1. Kapasitas Beban Kerja setiap karyawan tidak melebihi 20 Story Points. Batasan ini memastikan bahwa karyawan tidak kelebihan beban, yang dapat mempengaruhi produktivitas dan kesejahteraan.
2. Setiap tugas hanya boleh diberikan kepada tepat satu karyawan. Batasan ini mencegah duplikasi penugasan dan memastikan bahwa semua tugas dialokasikan secara eksklusif.
3. Setiap karyawan hanya boleh menerima tugas dari proyek yang sama. Batasan ini memastikan bahwa karyawan hanya akan berada dalam satu proyek sekaligus, yang secara tidak langsung hasil optimasi ini dapat memberikan informasi tentang pengalokasian tim proyek.

Objektif yang didefinisikan pada 3.2 konsisten dengan formulasi MOO yang akan dijelaskan pada 3.7.5. Batasan pada implementasi MOO sesuai dengan penjabaran diatas, kecuali:

1. Ketersediaan "Tugas" dan "Validitas Indeks" lebih relevan untuk RL (karena RL bersifat dinamis dan memerlukan validasi berulang), tetapi dalam MOO, ini dijamin oleh struktur model (penugasan unik dan definisi variabel).
2. "Kapasitas Beban Kerja" dalam MOO diterapkan melalui `max_workload`, yang dibatasi hingga 20, bukan langsung sebagai constraint ≤ 20 .

Untuk RL, objektif dan batasan ini dapat diterapkan dalam desain *reward function* dan environment, di mana WED menjadi komponen utama reward, variansi beban kerja dihitung untuk penalti, dan pemanfaatan karyawan idle didorong melalui prioritas penugasan

3.3 Pengumpulan Data

Tahap awal penelitian ini melibatkan persiapan data yang akan digunakan untuk melatih dan mengevaluasi model. Data didapatkan dari Direktorat Digital Business PT. Telkom Indonesia STO Kebayoran Baru yang terdiri dari dua jenis dataset, yaitu dataset karyawan dan dataset tugas.

Dataset pertama adalah dataset karyawan yang terdiri dari 109 karyawan. Seperti yang dicontohkan pada tabel 2, terdapat fitur nomor, ID karyawan, peran, lalu 65 metrik kompetensi. Dataset kedua adalah dataset tugas yang terdiri dari 300 tugas dari 5 proyek berbeda. Seperti yang direpresentasikan pada tabel 3, terdapat fitur ID tugas yang merupakan tugas tugas yang akan diberikan dan dikelompokkan dengan proyek, lima jenis ID proyek (P1, P2, P3, P4, P5), *story point* yang merupakan tolak ukur bobot kerja, dan 65 kebutuhan kompetensi yang sama dengan dataset karyawan. Kompetensi dituliskan pada dataset dengan menggabungkan kategori dan kompetensi dengan huruf titik.

Tabel 2. Tabel contoh fitur pada dataset *Employee*

No	Employee_id	Role	Mathematics.LinearAlgebra	...	MLOPS.Model Deployment
1	Employee_1	Data Analyst	3		1

Tabel 3. Tabel contoh fitur pada dataset *Task*

Task_id	Project_id	Story Point	Mathematics.LinearAlgebra	...	MLOPS.Model Deployment
Task_1	P1	3	4		5

Kedua dataset memiliki 65 metrik kompetensi yang sama. Kompetensi ini kemudian direpresentasikan dengan angka 0-5. Pembuat dataset tersebut pada PT. Telkom Indonesia STO Kebayoran Baru menginformasikan bahwa penilaian dilakukan dengan asesmen mandiri dan ditentukan dengan acuan yang didasarkan dari dokumen panduan *Competence Qualification Appraisal* (COQA) dari *Telkom Corporate University*. Isi dari dokumen tersebut tidak dapat penulis dapatkan dari pihak telkom, namun penilaian 0-5 secara terbuka diinformasikan dan telah

ditetukan oleh pemilik dataset dari PT. Telkom Indonesia STO Kebayoran Baru dengan acuan sesuai tabel 4.

Tabel 4. Tabel tolak ukur penilaian kompetensi

Skor	Tingkat Kompetensi	Deskripsi
0	<i>None</i>	Tidak memiliki pemahaman sama sekali.
1	<i>Novice</i>	Pemahaman dasar dengan kemampuan praktis minimal.
2	<i>Beginner</i>	Dapat melaksanakan tugas terkait dengan kompetensi dengan bimbingan atau dokumentasi ekstensif.
3	<i>Competent</i>	Mampu menangani tugas secara mandiri dengan tingkat kompleksitas sedang, sesekali memerlukan panduan.
4	<i>Proficient</i>	Mandiri sepenuhnya, mampu menangani tugas kompleks, membimbing junior, dan memecahkan masalah secara efektif.
5	<i>Expert</i>	Keahlian mendalam, mampu mengembangkan metode baru, memimpin proyek kompleks, dan membimbing secara ekstensif.

Kedua dataset tersebut memiliki metrik kompetensi sama dan dikelompokan menjadi 8 bidang, yang merupakan bidang matematika, statistika & probabilitas, struktur data & algoritma, *Machine Learning*, *Deep Learning*, *Data & Cloud Engineering*, *Machine Learning Operations*, dan terakhir Ekonometrika serta Analisis & Visualisasi Data. Kedelapan bidang tersebut dibagi menjadi 65 metrik kompetensi yang dijabarkan pada tabel 5.

Tabel 5. Tabel penjabaran kategori dan kompetensi

Kategori	Kompetensi
<i>Mathematics</i>	<i>Linear Algebra</i>
	<i>Differential Equations</i>
	<i>Optimization Technique</i>

	<i>Calculus</i>
	<i>Combinatorics & Graph</i>
<i>Statistics & Probabilities</i>	<i>Statistics</i>
	<i>Probability & Sampling</i>
	<i>Bayesian Method</i>
	<i>Hypothesis Testing</i>
	<i>A/B Testing</i>
	<i>CUPED</i>
<i>Data Structures & Algorithms</i>	<i>Complexity Analysis</i>
	<i>Programming</i>
	<i>Data Structures</i>
	<i>Algorithms</i>
	<i>Advanced Algorithms</i>
	<i>SQL</i>
<i>Econometrics, Data Analysis, and Data Visualization</i>	<i>Shell / Bash Scripting</i>
	<i>Data Preprocessing & EDA</i>
	<i>Data Viz & Storytelling</i>
	<i>Econometrics</i>
	<i>Regression Analysis</i>
	<i>Time Series Analysis</i>
	<i>Correlation Analysis</i>
	<i>Causal Analysis</i>
	<i>Dimensional Reduction (PCA)</i>
<i>Machine Learning</i>	<i>Supervised ML</i>
	<i>Unsupervised ML</i>
	<i>Reinforcement Learning</i>
	<i>Feature Engineering</i>
	<i>Ensemble Method</i>
<i>Deep Learning</i>	<i>Neural Network & DL</i>
	<i>NLP</i>

	<i>GAN</i>
	<i>Generative AI</i>
	<i>Computer VIision</i>
	<i>Audio & Speech Processing</i>
	<i>Transfer Learning</i>
	<i>Sequence Modelling</i>
	<i>Accelerated & Parallel Computing</i>
	<i>Edge Computing</i>
<i>Data & Cloud Engineering</i>	<i>Relational DB</i>
	<i>Non Relational DB</i>
	<i>Devops & CI/CD</i>
	<i>Data Pipeline</i>
	<i>Data Collection</i>
	<i>Data Wrangling</i>
	<i>Batch Processing</i>
	<i>Stream Processing</i>
	<i>Big Data Tech</i>
	<i>Data Warehousing</i>
	<i>ORM & REST API</i>
	<i>Microservices & Containerization</i>
	<i>Cloud Native</i>
	<i>Cloud & Serverless Architecture</i>
	<i>Data Security & Privacy</i>
	<i>Distributed Systems</i>
	<i>Monitoring, Logging, Alerting</i>
<i>MLOPS</i>	<i>ML Lifecycle</i>
	<i>ML Model Pipeline</i>
	<i>Model Versioning</i>
	<i>Automated Testing & Validation</i>

	<i>Scalable Model Training</i>
	<i>Model Deployment, Monitoring, Logging</i>

3.4 Eksplorasi Data

Eksplorasi data dilakukan untuk memahami karakteristik dataset dan memastikan kelayakan masalah penugasan tugas, baik untuk pendekatan MOO maupun RL. Analisis ini mencakup pemeriksaan kelayakan, distribusi beban kerja, karakteristik keterampilan, struktur proyek, dan dasar awal untuk evaluasi model.

3.4.1 Pemeriksaan Kelayakan Dataset

Pemeriksaan kelayakan dilakukan untuk memastikan bahwa 109 karyawan dapat menangani 300 tugas dengan batasan kapasitas maksimum 20 *story points* per karyawan, sebuah aspek yang penting untuk mendefinisikan kondisi selesai dalam RL dan memenuhi batasan dalam MOO.

Analisis ini dimulai dengan menghitung total kapasitas beban kerja, yaitu $109 \times 20 = 2.180$ *story point*, dan membandingkannya dengan total *story point* dari 300 tugas yang ada dalam dataset. Selain itu, rasio rata-rata tugas per karyawan dihitung sebagai $300/109 \approx 2,75$, memberikan gambaran awal tentang distribusi tugas. Prosesnya melibatkan penjumlahan *story points* dari semua tugas dan perbandingan dengan kapasitas total.

Hasil analisis ini memberikan wawasan penting. Jika total *story points* melebihi kapasitas, RL perlu dirancang untuk menangani tugas yang tidak dapat dialokasikan, sedangkan MOO akan menghadapi batasan yang lebih ketat dalam mencari solusi yang layak. Sebaliknya, jika total *story points* jauh di bawah kapasitas, banyak karyawan berpotensi menjadi menganggur, yang dapat mempengaruhi tujuan pemanfaatan karyawan dalam kedua pendekatan.

3.4.2 Distribusi Story Points

Analisis distribusi *story points* dilakukan untuk memahami variabilitas beban kerja dan dampaknya terhadap tujuan minimisasi variansi beban kerja (dengan bobot 0,1) serta kelayakan batasan kapasitas. Distribusi story points divisualisasikan melalui histogram yang didefinisikan pada rentang seperti 0-20, untuk melihat pola sebaran usaha yang diperlukan untuk menyelesaikan tugas. Selain itu, persentase tugas yang dapat dialokasikan dalam kelompok 20 *story point* dihitung untuk mengevaluasi seberapa banyak tugas yang dapat dialokasikan per karyawan pada kapasitas maksimum.

Hasil analisis ini memberikan wawasan bahwa distribusi *story points* yang lebar akan menyulitkan optimasi variansi beban kerja, karena tugas dengan story points tinggi dapat menyebabkan ketimpangan alokasi. Sebaliknya, distribusi yang sempit mempermudah penugasan tugas secara merata, yang dapat mempengaruhi penyesuaian bobot variansi dalam evaluasi model, terutama untuk memastikan keseimbangan antara tujuan optimasi.

3.4.3 Analisis Struktur Proyek

Analisis struktur proyek dilakukan untuk memahami distribusi tugas di antara lima proyek (P1 hingga P5) dan potensi pengaruhnya terhadap desain observasi RL, seperti penggunaan fitur spesifik proyek. Data tugas dikelompokkan berdasarkan project_id untuk menghitung jumlah tugas dan total story points per proyek, memberikan gambaran tentang sebaran beban kerja antar proyek.

Untuk RL, wawasan ini dapat dimanfaatkan dengan menambahkan fitur seperti mask proyek atau fitur spesifik proyek dalam desain observasi, sehingga agen dapat lebih baik dalam menangani perbedaan antara proyek dan meningkatkan generalisasi model.

3.5 Pra-pemrosesan Data

3.5.1 Pra-pemrosesan Umum

Pra-pemrosesan data umum dilakukan untuk menyiapkan dataset karyawan dan tugas agar konsisten dan sesuai dengan kebutuhan semua pendekatan optimasi. Langkah-langkah ini memastikan bahwa data karyawan dan tugas memiliki format yang konsisten, bebas dari anomali, dan dilengkapi dengan metrik kecocokan keterampilan yang relevan untuk mendukung optimasi. Berikut adalah langkah-langkah pra pemrosesan data umum.

Tabel 6. Tabel Tahap Preprocessing Umum

Langkah	Deskripsi	Output
Identifikasi Kolom Keterampilan	Mengidentifikasi 65 kolom keterampilan, mengabaikan kolom non-keterampilan seperti employee_id, task_id, project_id, story_points, dan Role.	Daftar 65 kolom keterampilan (contoh: Mathematics.LinealAlgebra, MLOPS.Model Deployment).
Pembersihan Data	Memeriksa dan menangani nilai hilang (missing values) dengan pengisian nol untuk keterampilan yang tidak tersedia.	DataFrame tanpa nilai hilang.
Perhitungan WED	Menghitung Weighted Euclidean Distance untuk setiap pasangan karyawan-tugas. Hanya keterampilan dengan task_skill > 0 dipertimbangkan.	Matriks WED (109 × 300).

3.5.1 Pra-pemrosesan *Greedy*

Pendekatan *Greedy* memerlukan pra pemrosesan data yang sederhana namun efektif untuk mendukung algoritma berbasis kecocokan keterampilan secara sekuensial. Proses ini bertujuan untuk menyiapkan data karyawan dan tugas dalam format yang memungkinkan iterasi cepat untuk penugasan tugas berdasarkan WED terkecil. Selain langkah umum, pra pemrosesan untuk *Greedy* mencakup normalisasi keterampilan, pengacakan urutan tugas, dan inisialisasi struktur penugasan. Pendekatan ini tidak memberlakukan batasan proyek secara ketat, memungkinkan karyawan untuk ditugaskan ke tugas dari proyek berbeda demi memaksimalkan kecocokan keterampilan.

Tabel 7. Tabel Tahap Preprocessing Pendekatan *Greedy*

Langkah	Deskripsi	Output
Normalisasi Keterampilan	Membagi nilai keterampilan (0-5) dengan 5.0 untuk skala 0-1.	Matriks keterampilan: emp_skills (109×65), task_skills (300×65).
Pengacakan Urutan Tugas	Mengacak indeks tugas menggunakan np.random.permutation untuk menghindari bias urutan.	Array indeks tugas acak (ukuran 300).
Inisialisasi Struktur Penugasan	Membuat dictionary assignments dan array workload untuk melacak penugasan dan beban kerja.	Dictionary assignments (109 kunci), array workload (109).

3.5.1 Pra-pemrosesan MOO

Pendekatan ini memerlukan pra pemrosesan data yang mendukung pembangunan model matematis. Proses ini mencakup pembersihan data spesifik, pengelompokan tugas berdasarkan proyek, dan penyimpanan story points serta similarity scores dalam struktur yang sesuai. Selain langkah umum, pra pemrosesan untuk MOO melibatkan penghapusan kolom tidak relevan, pembuatan dictionary untuk tugas per proyek, dan perhitungan similarity score berbasis WED untuk digunakan dalam fungsi objektif.

Tabel 8. Tabel Tahap Preprocessing Pendekatan MOO

Langkah	Deskripsi	Output
Pengelompokan Tugas per Proyek	Membuat dictionary company_tasks yang mengelompokkan tugas berdasarkan project_id.	Dictionary company_tasks dengan 5 kunci (P1-P5).
Ekstraksi Story Points dan Project ID	Menyimpan story_points sebagai dictionary untuk setiap tugas per proyek.	Dictionary story_points (300 pasangan tugas-story points).
Perhitungan Similarity Score	Mengubah WED menjadi similarity score menggunakan $1 / (1 + \text{WED})$ untuk setiap pasangan karyawan-tugas.	Matriks similarity score (109×300).
Penyusunan Data untuk Model	Menyiapkan data dalam format matriks yang kompatibel dengan Gurobi, termasuk indeks karyawan, tugas, dan proyek.	Struktur data untuk variabel keputusan $x[i,j,k]$, $y[j,k]$. dan \max_{workload} .

3.5.1 Pra-pemrosesan DQN

Pendekatan *Deep Q-Network* (DQN) berbasis *Reinforcement Learning* (RL) memerlukan pra pemrosesan data yang mendukung lingkungan simulasi dinamis (TaskAssignmentEnv). Proses ini mencakup normalisasi keterampilan, padding data untuk konsistensi dimensi, pemetaan project ID, dan konversi data ke tensor PyTorch untuk efisiensi komputasi pada GPU. Selain langkah umum, pra pemrosesan untuk DQN melibatkan pembuatan matriks state awal berbasis top-5 skor WED, yang digunakan sebagai bagian dari ruang observasi untuk pelatihan model.

Tabel 9. Tabel Tahap Preprocessing Pendekatan DQN

Langkah	Deskripsi	Output
Normalisasi Keterampilan	Membagi nilai keterampilan (0-5) dengan 5.0 untuk skala 0-1.	Matriks keterampilan: task_skills (300×65), employee_skills (109×65).
Padding Data	Menambahkan padding nol hingga max_tasks=500 dan max_employees=200 untuk konsistensi dimensi	Tensor: task_skills (500×65), employee_skills (200×65).
Pemetaan Project ID	Memetakan project_id ke indeks numerik (1-5) untuk efisiensi.	Tensor project_ids (500).
Konversi ke Tensor	Mengonversi data keterampilan, story points, dan project IDs ke	Tensor: task_skills, employee_skills,

	tensor PyTorch di perangkat (CPU/GPU)	story_points (500), project_ids.
Inisialisasi State Matrix	Menghitung similarity matrix berbasis WED, mengambil top-5 similarity scores per tugas untuk state_matrix.	Tensor state_matrix (500×5).

3.6 Pendekatan *Greedy*

Pendekatan *greedy* digunakan sebagai metode dasar untuk menetapkan performa awal yang dapat menjadi pembanding bagi pendekatan RL dan MOO dalam penugasan tugas kepada karyawan. Pendekatan ini mengalokasikan setiap tugas kepada karyawan dengan skor *Weighted Euclidean Distance* (WED) terbaik, dengan mematuhi batasan kapasitas maksimum 20 story points per karyawan. Proses penugasan dilakukan secara berurutan untuk setiap tugas tanpa mempertimbangkan batasan proyek, sehingga karyawan dapat mengambil tugas dari berbagai proyek selama kapasitasnya memungkinkan. Setelah penugasan selesai, metrik evaluasi dihitung untuk menilai kinerja pendekatan ini, meliputi: rata-rata skor WED untuk mengevaluasi kesesuaian keterampilan, variansi beban kerja untuk menilai distribusi beban kerja antar karyawan, dan jumlah karyawan idle untuk mengukur pemanfaatan tenaga kerja.

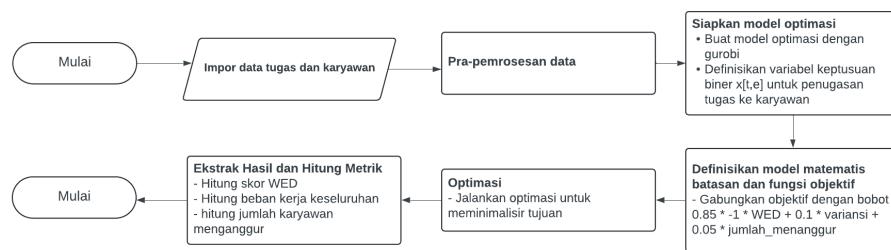
Hasil dari pendekatan ini memberikan gambaran awal tentang penugasan yang berfokus pada kecocokan keterampilan tanpa mempertimbangkan keseimbangan beban kerja atau pemanfaatan karyawan secara optimal.

3.7 Pengembangan Pendekatan MOO

Dataset yang digunakan terdiri dari 109 karyawan dan 300 tugas, di mana setiap tugas termasuk dalam salah satu dari lima proyek (P1 hingga P5). Setiap karyawan dan tugas memiliki vektor keterampilan 65 dimensi dengan nilai antara 0 hingga 5, serta tugas memiliki *story points* sebagai ukuran usaha, yang berkisar

dari 1 hingga 8 berdasarkan distribusi data. Pendekatan MOO ini bertujuan untuk mengoptimalkan penugasan tugas dengan memastikan bahwa setiap tugas hanya diberikan kepada satu karyawan, beban kerja karyawan tidak melebihi kapasitas maksimum 20 poin cerita, dan tujuan-tujuan optimasi tercapai sesuai bobot yang ditetapkan, yaitu 0,85 untuk kesesuaian keterampilan, 0,1 untuk variansi beban kerja, dan 0,05 untuk jumlah karyawan idle.

Formulasi matematis untuk pendekatan MOO dirancang untuk merepresentasikan permasalahan penugasan tugas secara sistematis. Model ini mencakup definisi himpunan, parameter, variabel keputusan, kendala, dan fungsi tujuan, yang semuanya dijelaskan secara rinci untuk memastikan kejelasan dan ketepatan dalam implementasi.



Gambar 9. Flowchart Pendekatan MOO

3.7.1 Himpunan dan Indeks

Himpunan dan indeks berikut digunakan untuk mendefinisikan elemen-elemen dalam model:

- E : Himpunan karyawan (*Employees*), dimana $e \in \{0, \dots, 108\}$, dengan $|E| = 109$. Himpunan ini merepresentasikan seluruh karyawan yang tersedia untuk diberikan tugas, dengan setiap karyawan memiliki identifikasi unik dari 0 hingga 108.
- T : Himpunan Tugas (*Task*), dimana $t \in \{0, \dots, 299\}$, dengan $|T| = 300$. Himpunan ini mencakup semua tugas yang harus diberikan

kepada karyawan, dengan setiap tugas memiliki kebutuhan keterampilan dan usaha tertentu.

- P : Himpunan proyek (*Project*), dimana $p \in \{P1, P2, P3, P4, P5\}$, dengan $|P|= 5$. Proyek-proyek ini mengelompokkan tugas-tugas berdasarkan struktur organisasi, dimana setiap proyek memiliki subset tugas yang berbeda.
- T_p : Subset tugas yang termasuk dalam proyek p , dimana $T_p \subseteq T$, dengan $\bigcup_{p \in P} T_p = T$ dan $T_p \cap T_{p'} = \emptyset$ untuk $p \neq p'$.
- K : Himpunan dimensi keterampilan, dimana $k \in K = \{1, 2, \dots, 65\}$, dengan $|K|=65$. Setiap dimensi mewakili jenis keterampilan tertentu, seperti “Mathematics.Linear Algebra” atau ”MLOPS.Ethical AI & Bias Mitigation”.

3.7.2 Parameter

Parameter berikut diambil dari dataset untuk mendukung perhitungan dalam model:

- $S_{e,k}$: Tingkat keterampilan (*Skill*) karyawan e pada dimensi keterampilan k , dimana $S_{e,k} \in \{0, 1, 2, 3, 4, 5\}$. Parameter ini menunjukkan kemampuan karyawan dalam suatu keterampilan, misalnya $S_{0,1} = 3$ berarti karyawan 0 memiliki keterampilan 3 pada keterampilan urutan 1 (Mathematics.Linear Algebra).
- $R_{t,k}$: Kebutuhan (*Requirement*) keterampilan tugas t pada dimensi keterampilan k , dimana $R_{t,k} \in \{0, 1, 2, 3, 4, 5\}$. Parameter ini menunjukkan tingkat keterampilan yang diperlukan untuk menyelesaikan tugas, misalnya $R_{0,1} = 3$ berarti tugas indeks 0 membutuhkan keterampilan 3 pada keterampilan urutan 1 (Mathematics.Linear Algebra).

- SP_t : Singkatan dari *Story Point* tugas t, dimana $SP_t \in \{1, 2, 3, 5, 8\}$
 - . *Story Point* merupakan usaha yang diperlukan untuk menyelesaikan yang memiliki rentang angka fibonacci dari 1 hingga 8.
 - C_{max} : Kapasitas maksimum beban kerja per karyawan, dimana $C_{max} = 20$. Nilai ini menetapkan batas atas beban kerja untuk mencegah kelebihan beban pada karyawan.

$$\bullet \quad WED(e, t) = 1 - \frac{\sqrt{\sum_{k \in K} w_{e,t,k} (S_{e,k} - R_{t,k})^2}}{max_{wed}} \dots \dots \dots (15)$$

dimana:

- $w_{e,t,k} = \frac{1}{1+\alpha \max(0, S_{e,k} - R_{t,k})}$, dengan $\alpha = 0.5$
 - $\text{max_wed} = \sqrt{\sum_{k \in K} \left(\frac{1}{1+\alpha \cdot 5}\right) (0 - 5)^2}$, yaitu nilai maksimum WED ketika semua keterampilan berbeda sebesar 5.
 - $WED(e, t) \in [0, 1]$, dengan nilai lebih tinggi menunjukkan kecocokan keterampilan yang lebih baik. Metrik ini dirancang untuk memberikan penalti lebih kecil pada kelebihan keterampilan ($S_{e,k} < R_{t,k}$), yang mencerminkan preferensi praktis dalam penugasan tugas.

3.7.3 Variabel Keputusan (*Decision Variables*)

Variabel keputusan dalam model ini dirancang untuk merepresentasikan keputusan penugasan tugas kepada karyawan, serta untuk mendukung perhitungan metrik tambahan seperti beban kerja dan pemanfaatan karyawan. Model ini menggunakan pendekatan pemrograman linear dengan optimasi multi-tujuan untuk mengoptimalkan

penugasan 300 tugas kepada 109 karyawan, yang dikelompokkan ke dalam 5 proyek (P1 hingga P5).

Variabel keputusan berikut didefinisikan untuk memastikan bahwa solusi yang dihasilkan dapat mencerminkan keputusan penugasan serta memenuhi tujuan optimasi yang telah ditetapkan, yaitu memaksimalkan kesesuaian keterampilan, meminimalkan variansi beban kerja, dan meminimalkan jumlah karyawan yang tidak diberikan tugas.

- $x_{t,e,p}$: variabel biner, dimana $x_{t,e,p} = 1$ jika tugas $t \in T_p$ diberikan kepada karyawan e , dan 0 jika tidak. Variabel ini merupakan inti dari keputusan penugasan tugas dalam model, yang menentukan alokasi tugas kepada karyawan dengan mempertimbangkan struktur proyek. Formulasi variabel ini adalah:

$$x_{t,e,p} \in \{0, 1\} \quad \forall e \in E, \forall p \in P, \forall t \in T_p \dots \dots \dots \quad (16)$$

Jumlah variabel keputusan $x_{t,e,p}$ adalah $|T| \times |E| = 300$ tugas x 109 karyawan = 32.700.

- $y_{e,p}$: variabel biner, dimana $y_{e,p} = 1$ jika karyawan e diberikan setidaknya satu tugas dari proyek p , dan 0 jika tidak. Variabel ini digunakan untuk melacak apakah karyawan memiliki pekerjaan atau tidak. Variabel ini memungkinkan model untuk mengetahui apakah seorang karyawan terlibat dalam proyek tertentu, sehingga mendukung minimisasi jumlah karyawan idle. Formulasi variabel ini adalah:

$$y_{e,p} \in \{0, 1\} \quad \forall e \in E, \forall p \in P \dots \dots \dots \quad (17)$$

Jumlah variabel keputusan $y_{e,p}$ adalah $|E| \times |P| = 109$ karyawan x 5 proyek = 545.

- **max_workload:** Variabel bilangan bulat yang menyatakan beban kerja maksimum di antara semua karyawan, digunakan sebagai proksi untuk mengurangi variansi beban kerja. Variabel ini didefinisikan dengan batasan bawah 0 dan batasan atas $C_{max} = 20$, sesuai dengan kapasitas maksimum beban kerja per karyawan yang telah ditetapkan sebelumnya. Variabel ini memungkinkan model untuk meminimalkan beban kerja maksimum sebagai cara tidak langsung untuk menyeimbangkan distribusi beban kerja di antara karyawan. Hanya ada satu keputusan variabel max_workload, yaitu max_workload=20.

$$\text{max_workload} \in \{0, 1, \dots, C_{\max}\}, \quad C_{\max} = 20 \quad \dots \dots \dots \quad (18)$$

Jumlah variabel keputusan senilai 33.246, memiliki beberapa implikasi penting dalam konteks optimasi. Pertama, dengan 33.246 variabel keputusan, masalah ini menjadi cukup besar untuk diselesaikan menggunakan *Linear Programming*. Gurobi harus mengevaluasi kombinasi yang sangat banyak untuk menemukan solusi optimal, yang meningkatkan waktu komputasi, terutama pada dataset dengan skala 109 karyawan dan 300 tugas.

Setiap variabel keputusan membutuhkan ruang memori untuk menyimpan nilainya selama proses optimasi. Dengan 33.246 variabel, kebutuhan memori menjadi signifikan, terutama karena Gurobi juga menyimpan informasi tambahan seperti matriks kendala dan cabang dalam algoritma *branch-and-bound*. Hal ini dapat menjadi tantangan jika optimasi dilakukan pada perangkat dengan sumber daya terbatas.

Jumlah variabel keputusan sebanding dengan $|T| \times |E|$, sehingga jika skala masalah meningkat (misalnya, 1.000 karyawan dan 3.000 tugas), jumlah variabel akan meningkat secara kuadratik, membuat pendekatan MOO ini kurang praktis untuk masalah yang sangat besar. Inilah salah satu

alasan mengapa pendekatan *Reinforcement Learning* (RL) dipertimbangkan, karena RL dapat menawarkan solusi yang lebih cepat meskipun dengan sedikit pengorbanan pada optimalitas.

Banyaknya variabel $x_{t,e,p}$ (32.700) menunjukkan bahwa model ini sangat rinci dalam memodelkan penugasan, tetapi juga berpotensi memiliki redundansi. Misalnya, batasan "satu proyek per karyawan" dapat mengurangi ruang solusi, tetapi tidak mengurangi jumlah variabel yang harus didefinisikan. Penggunaan variabel tambahan seperti $y_{e,p}$ juga menambah kompleksitas, meskipun membantu dalam perhitungan tujuan (misalnya, karyawan *idle*).

3.7.4 Batasan (*Constraints*)

Batasan berikut diterapkan untuk memastikan bahwa penugasan tugas memenuhi syarat-syarat kelayakan dan konsistensi dalam model optimasi. Setiap batasan diformulasikan secara matematis dengan notasi yang konsisten dengan definisi sebelumnya, untuk memberikan kejelasan dalam implementasi model.

- Setiap tugas hanya dapat diberikan kepada satu karyawan. Batasan ini memastikan bahwa setiap tugas hanya diberikan sekali, mencegah tugas yang tidak diberikan atau diberikan lebih dari sekali. Dengan $|T| = 300$, batasan ini menghasilkan 300 persamaan, satu untuk setiap penugasan. Formulasi batasan ini adalah sebagai berikut:

- Variabel $y_{e,p}$ harus bernilai 1 jika karyawan e diberikan setidaknya satu tugas dari proyek p, dan 0 jika tidak. Batasan ini diterapkan

menggunakan dua persamaan untuk memastikan hubungan logis antara variabel x dan y.

$$\sum_{t \in T_p} x_{t,e,p} \geq 1 - (1 - y_{e,p}) \quad \forall e \in E, \forall p \in P \dots\dots\dots (20)$$

$$\sum_{t \in T_p} x_{t,e,p} \leq |T_p| \cdot y_{e,p} \quad \forall e \in E, \forall p \in P \dots\dots\dots (21)$$

Persamaan pertama memastikan bahwa jika ada setidaknya satu tugas dari proyek p yang diberikan kepada karyawan e, maka $y_{e,p}$ harus bernilai 1. Persamaan kedua memastikan bahwa jika tidak ada tugas yang diberikan, maka $y_{e,p}$ harus bernilai 0, dengan $|T_p|$ sebagai jumlah tugas dalam proyek p.

- Satu Proyek per karyawan. Setiap Karyawan hanya boleh diberikan tugas dari proyek yang sama. Batasan ini membatasi karyawan untuk hanya terlibat dalam satu proyek, sehingga mencegah karyawan bekerja pada beberapa proyek secara bersamaan. Hal ini mencerminkan asumsi bahwa karyawan harus fokus pada satu proyek untuk efisiensi organisasi. Kendala ini menghasilkan 109 persamaan, satu untuk setiap karyawan dalam himpunan E. Formulasi batasan ini adalah sebagai berikut:

$$\sum_{p \in P} y_{e,p} \leq 1 \quad \forall e \in E \dots\dots\dots (22)$$

- Kapasitas beban kerja setiap karyawan adalah 20 *story points*. Total *story point* tidak boleh melebihi kapasitas maksimum. Dengan $C_{max} = 20$, batasan ini mencegah kelebihan beban pada karyawan. Karena $SP_t \in \{1, 2, 3, 5, 8\}$, seorang karyawan dapat menangani 2 hingga 5 tugas (misalnya, $20/8 \approx 2.5$, atau hingga

$20/1 = 20$). Batasan ini menghasilkan 109 persamaan. Formulasi batasan ini adalah sebagai berikut:

$$\sum_{t \in T_p} SP_t \cdot x_{t,e,p} \leq \text{max_workload} \quad \forall e \in E, \forall p \in P \dots \dots (23)$$

- Variabel `max_workload` harus lebih besar atau sama dengan total beban kerja setiap karyawan di semua proyek. Batasan ini memastikan bahwa `max_workload` mencerminkan beban kerja maksimum di antara semua karyawan, yang kemudian digunakan untuk mengurangi variansi beban kerja melalui minimalisasi. Batasan ini menghasilkan 109 persamaan, satu untuk setiap karyawan dalam himpunan E. Formulasi batasan ini adalah sebagai berikut:

$$\text{max_workload} \geq \sum_{p \in P} \sum_{t \in T_n} SP_t \cdot x_{t,e,p} \quad \forall e \in E \dots \dots (24)$$

3.7.5 Fungsi Objektif (*Objective Function*)

Pendekatan Multi-Objective Optimization (MOO) dalam penelitian ini mengoptimalkan tiga tujuan utama: memaksimalkan kesesuaian keterampilan antara karyawan dan tugas, meminimalkan variansi beban kerja di antara karyawan, serta meminimalkan jumlah karyawan yang tidak diberikan tugas.

Objektif pertama adalah meminimalisir jumlah karyawan yang tidak diberikan tugas, sehingga memaksimalkan pemanfaatan tenaga kerja yang dapat diformulasikan sebagai berikut.

Jika karyawan e diberikan tugas dari setidaknya satu proyek, maka

$\sum_{p \in P} y_{e,p} = 1$, sehingga $1 - \sum_{p \in P} y_{e,p} = 0$. Sebaliknya, jika karyawan tidak

diberikan tugas sama sekali, maka $\sum_{p \in P} y_{e,p} = 0$, dan kontribusi karyawan

tersebut ke Obj_1 adalah 1. Dengan $|E| = 109$, nilai Obj_1 berada dalam rentang $[0,109]$. Objektif ini memastikan memastikan bahwa sebanyak mungkin karyawan dimanfaatkan dalam penugasan tugas, yang penting untuk efisiensi organisasi, terutama mengingat rasio tugas per karyawan yang tinggi (sekitar 2,75 tugas per karyawan).

Objektif kedua adalah memaksimalkan total skor kesesuaian keterampilan, yang diukur menggunakan metrik WED, di seluruh penugasan tugas. Objektif ini bertujuan untuk memaksimalkan skor total WED, yang merefleksikan seberapa baik pemberian tugas diberikan kepada karyawan paling tepat. Metrik WED(e,t), yang telah didefinisikan sebelumnya, berada dalam rentang [0,1], dengan nilai yang lebih tinggi menunjukkan kecocokan keterampilan yang lebih baik antara karyawan dan tugas t . Dengan total $|T|=300$ tugas, nilai Obj1 berada dalam rentang [0,300]. Tujuan ini mencerminkan prioritas utama penelitian untuk memastikan bahwa karyawan yang ditugaskan memiliki keterampilan yang paling sesuai dengan kebutuhan tugas, sehingga meningkatkan efisiensi dan kualitas penyelesaian tugas. Definisi objektif pertama adalah sebagai berikut:

Objektif ketiga adalah meminimalisir variansi beban kerja. Objektif kedua adalah meminimalkan variansi atau standar deviasi beban kerja antara karyawan. Secara ideal, variansi beban kerja dapat didefinisikan sebagai:

$$Variansi = \frac{1}{|E|} \sum_{e \in E} (W_e - W')^2 \dots \dots \dots \quad (27)$$

$$\text{dimana } W_e = \sum_{p \in P} \sum_{t \in T_p} SP_t \cdot x_{t,e,p} \text{ dan } W' = \frac{1}{|E|} \sum_{e \in E} W_e$$

Namun, formulasi ini bersifat non-linear karena adanya kuadrat $(W_e - W')^2$, yang tidak dapat dioptimalkan secara langsung dalam kerangka pemrograman linear. Untuk menjaga model tetap linear, tujuan ini diubah menjadi meminimalkan beban kerja maksimum sebagai proksi:

Variabel $\max_workload$ telah didefinisikan sebelumnya sebagai beban kerja maksimum di antara semua karyawan, dengan rentang $[0, C_{\max}]$, di mana $C_{\max} = 20$. Pendekatan ini dipilih karena meminimalkan $\max_workload$ secara tidak langsung mendorong distribusi beban kerja yang lebih merata, dengan mencegah adanya karyawan yang memiliki beban kerja terlalu tinggi.

Proses transformasi ini melibatkan pengenalan variabel `max_workload` dan batasan (*constraint*) yang memastikan bahwa `max_workload` lebih besar atau sama dengan beban kerja setiap karyawan, sehingga tujuan tetap linear dan dapat dioptimalkan menggunakan solver pemrograman linear. Pendekatan ini dipilih untuk menjaga efisiensi komputasi, mengingat skala masalah yang besar, dan karena minimisasi beban kerja maksimum merupakan proksi yang umum digunakan dalam literatur optimasi untuk menyeimbangkan beban kerja.

Untuk mengintegrasikan ketiga tujuan dalam pendekatan MOO, digunakan metode weighted sum dengan normalisasi untuk memastikan bahwa setiap tujuan berkontribusi secara proporsional meskipun memiliki

skala yang berbeda. Bobot yang diterapkan mencerminkan prioritas penelitian dan selaras dengan bobot yang digunakan dalam salah satu pendekatan *Reinforcement Learning* (RL) sebagai pembanding.

Solver optimasi yang digunakan adalah Gurobi. Pada kasus ini, Objektif pertama akan dimaksimalkan, sedangkan objektif kedua dan ketiga akan diminimalisir. Sehingga, perhitungan objektif MOO adalah sebagai berikut:

$$\begin{aligned} \text{MOO Objective} &= -w_1 \cdot Obj_1 + w_2 \cdot Obj_2 + w_3 \cdot Obj_3 \\ &= 0.1 \cdot Obj_1 - 0.85 \cdot Obj_2 + 0.05 \cdot Obj_3 \dots (29) \end{aligned}$$

Bobot ini mencerminkan prioritas penelitian, di mana kesesuaian keterampilan menjadi fokus utama (0,85), diikuti oleh keseimbangan beban kerja (0,1) dan pemanfaatan karyawan (0,05). Normalisasi memastikan bahwa setiap tujuan memiliki dampak yang seimbang dalam fungsi tujuan gabungan, meskipun skala nilainya berbeda.

3.7.6 Implementasi dan Relevansi

Model MOO ini diimplementasikan menggunakan Gurobi, sebuah solver optimasi yang mampu menangani masalah pemrograman linear integer dengan skala besar, untuk mendapatkan solusi optimal atau mendekati optimal pada dataset utama yang terdiri dari 109 karyawan dan 300 tugas. Total variabel keputusan dalam model ini adalah 33.246, yang terdiri dari 32.700 variabel $x_{t,e,p}$, 545 variabel $y_{e,p}$ dan 1 variabel max_workload.

Jumlah variabel keputusan yang besar ini menunjukkan kompleksitas model, dengan implikasi signifikan terhadap kebutuhan komputasi. Proses optimasi memerlukan evaluasi terhadap kombinasi solusi yang sangat banyak, yang dapat meningkatkan waktu komputasi

hingga beberapa menit atau bahkan jam, tergantung pada spesifikasi perangkat keras dan pengaturan parameter solver. Selain itu, kebutuhan memori untuk menyimpan variabel-variabel ini, bersama dengan matriks kendala, menjadi tantangan tersendiri, terutama pada perangkat dengan sumber daya terbatas. Skalabilitas model juga menjadi perhatian, karena jumlah variabel keputusan meningkat secara kuadratik dengan bertambahnya jumlah karyawan dan tugas, membuat pendekatan ini kurang praktis untuk dataset yang sangat besar.

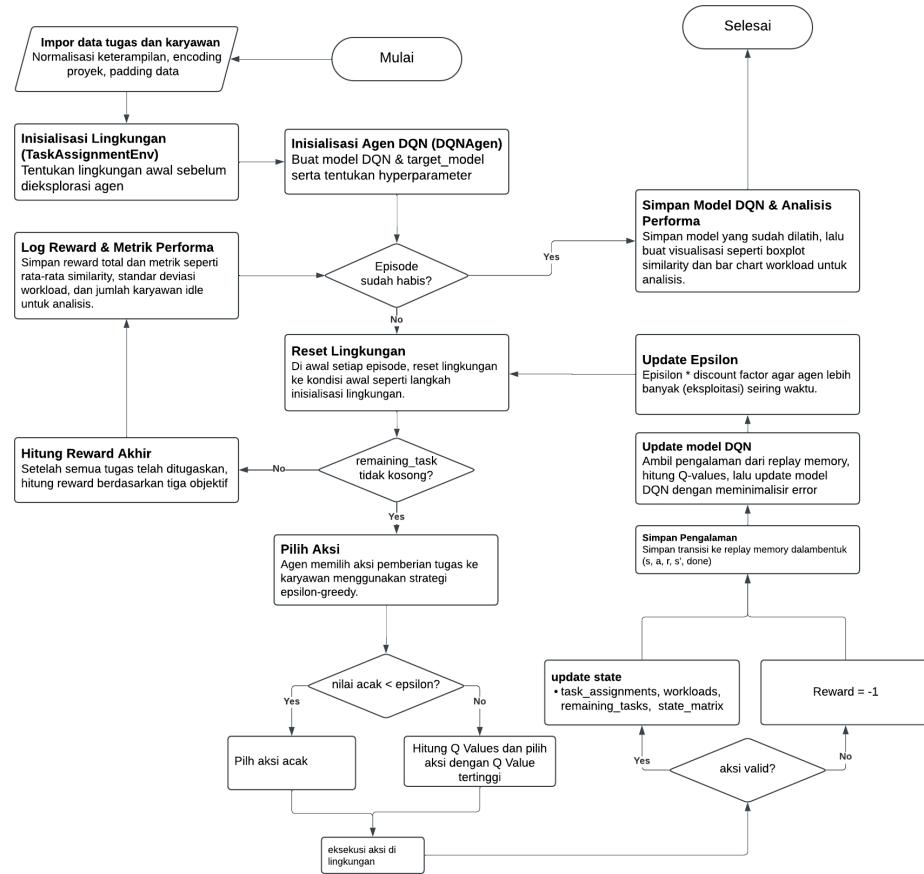
Berdasarkan analisis awal, solusi dari MOO diharapkan menghasilkan skor WED rata-rata sekitar 0,85 hingga 0,90 per tugas, yang menunjukkan tingkat kesesuaian keterampilan yang tinggi, sejalan dengan distribusi keterampilan yang cukup seragam pada dataset. standar deviasi beban kerja diperkirakan berada di kisaran 2 hingga 3 poin cerita, dengan nilai max_workload sekitar 10 hingga 12, menunjukkan distribusi beban kerja yang seimbang di antara karyawan. Jumlah karyawan idle diharapkan sangat kecil, antara 0 hingga 5, mengingat rasio tugas per karyawan (300 tugas untuk 109 karyawan, atau sekitar 2,75 tugas per karyawan) memungkinkan pemanfaatan tenaga kerja yang tinggi.

Solusi MOO ini berfungsi sebagai acuan untuk mengevaluasi pendekatan RL yang dikembangkan dalam penelitian ini. RL diharapkan mencapai performa yang mendekati MOO, dengan skor WED rata-rata di atas 0,80, standar deviasi beban kerja sekitar 3 hingga 4, dan jumlah karyawan idle antara 5 hingga 10. Namun, keunggulan utama RL terletak pada efisiensi komputasinya, di mana RL dapat menyelesaikan penugasan dalam hitungan detik, dibandingkan dengan MOO yang memerlukan waktu lebih lama karena kompleksitas modelnya. Pendekatan MOO memberikan dasar yang kuat untuk analisis optimalitas, sementara RL menawarkan solusi yang lebih praktis untuk aplikasi dunia nyata dengan kebutuhan komputasi yang lebih efisien.

3.8 Pengembangan DQN

3.8.1 Gambaran Umum

Pengembangan DQN mencakup pra pemrosesan data untuk merepresentasikan state dan action, pembuatan lingkungan simulasi untuk mendefinisikan dinamika penugasan, perancangan fungsi reward dan batasan untuk mendukung objektif berganda, pembangunan model *neural network* untuk memprediksi *Q-values*, serta pelatihan model menggunakan strategi epsilon-greedy dan replay memory. Dataset yang digunakan diproses sebagai tensor untuk mendukung komputasi efisien dengan GPU.



Gambar 10. Gambaran Umum Proses DQN

3.8.2 Representasi Lingkungan dengan *State* dan *Action*

Representasi *state* dan *action* merupakan dasar pengembangan DQN, yang menentukan bagaimana informasi penugasan tugas diolah dalam lingkungan RL. State mencakup matriks WED, vektor beban kerja, fitur global, dan mask tugas, semuanya direpresentasikan sebagai tensor untuk efisiensi komputasi. Ruang aksi mendefinisikan keputusan penugasan atau pemindahan tugas, memungkinkan model untuk bereaksi terhadap dinamika lingkungan.

Ruang observasi direpresentasikan dalam dictionary tensor yang mendefinisikan semua input yang diterima agen dari lingkungan pada setiap langkah pelatihan. Terdapat empat komponen pada ruang observasi, yaitu *state_matrix*, *workload_matrix*, *global_features*, dan *task_mask*.

Tensor pertama, yang disebut *state_matrix*, adalah matriks berukuran 500×5 yang berfungsi untuk menyimpan 5 nilai WED tertinggi untuk masing-masing tugas. Tensor ini memungkinkan model untuk fokus pada kandidat karyawan terbaik untuk setiap tugas, dengan hanya 300 tugas aktif dan sisanya ditambahkan *padding* dengan nol untuk konsistensi dimensi. Formulasi perhitungan WED dapat dilihat di bagian 2.1.8. Representasi tensor *state_matrix* dapat dilihat sebagai berikut. Kolom Task_ID menunjukkan indeks tugas (T1 hingga T500), kolom Rank1 hingga Rank5 berisi nilai kecocokan untuk lima karyawan dengan WED terbaik untuk tugas tersebut, baris T1 hingga T300 berisi nilai aktif, sedangkan T301 hingga T500 diisi nol karena hanya 300 tugas aktif.

Tabel 10. Representasi *state_matrix*

Task_ID	rank1	rank2	rank3	rank4	rank5
T1	0.95	0.90	0.85	0.80	0.75
...

T300	0.92	0.88	0.86	0.77	0.53
T301	0	0	0	0	0
...
T500	0	0	0	0	0

Tensor kedua, yaitu *workload_matrix*, adalah vektor berukuran 200 yang menyimpan beban kerja untuk setiap karyawan. Tensor ini mencerminkan distribusi beban kerja saat ini, memungkinkan model untuk mengevaluasi keseimbangan beban kerja dan mencegah kelebihan kapasitas. Hanya 109 elemen pertama aktif, dengan sisanya ditambahkan padding dengan nol. Kolom Employee_ID menunjukkan indeks karyawan (Talent 1 hingga Talent 200), kolom Workload berisi jumlah story points yang ditugaskan ke karyawan (contoh: 15.0, 8.0, dll.), dan baris Talent 1 hingga Talent 109 berisi nilai aktif, sedangkan Talent 110 hingga Talent 200 diisi nol karena hanya 109 karyawan aktif.

Tabel 11. Representasi *workload_matrix*

Employee_ID	Workload
Talent 1	15.0
...	...
Talent 109	12.0
Talent 110	0.0
...	...
Talent 200	0.0

Tensor ketiga, yaitu *global features*, adalah vektor berukuran 2 yang berfungsi untuk memberikan informasi kontekstual tentang status lingkungan, yaitu proporsi tugas yang belum ditugaskan dan proporsi karyawan idle. Tensor ini diformulasikan sebagai:

$$G = [p_t, p_i], p_t = \frac{|tugas\ tersisa|}{jumlah\ tugas}, p_i = \frac{\sum_{j=1}^{maxtugas} 1(W_j=0)}{maxtugas} \dots\dots\dots(30)$$

keterangan:

- $G \in \mathbb{R}^2$: Vektor fitur global
- p_t : proporsi tugas yang belum ditugaskan
- p_i : Proporsi karyawan menganggur
- $1(W_j = 0)$: Fungsi indikator, 1 jika $W_j = 0$, dan 0 sebaliknya

Tensor ini direpresentasikan pada tabel berikut. Baris Proporsi Tugas Tersedia menunjukkan p_t misalnya 0.60 berarti 60% tugas (180 dari 300) belum ditugaskan. Baris Proporsi Karyawan Menganggur menunjukkan p_i , misalnya 0.20 berarti 20% karyawan (22 dari 109) memiliki beban kerja nol.

Tabel 12. Representasi *global_features*

Fitur	Nilai
Proporsi Tugas Tersedia	0.60
Proporsi Karyawan Menganggur	0.20

Tensor keempat, yaitu *task_mask*, adalah vektor biner berukuran 500 500 500 yang menunjukkan ketersediaan tugas, dengan $M_i = 1$ jika tugas i tersedia untuk ditugaskan dan $M_i = 0$ jika sudah ditugaskan atau

tidak aktif. Tensor ini memandu model untuk hanya memilih tugas yang valid, dengan hanya 300 elemen pertama relevan dan sisanya bernilai nol. Representasi tensor *task_mask* dapat dilihat pada tabel berikut. Kolom Task_ID menunjukkan indeks tugas (T1 hingga T500). Kolom ketersediaan berisi nilai biner: 1 jika tugas tersedia, 0 jika tidak. Baris T1 hingga T300 berisi nilai aktif (1 atau 0 tergantung status), sedangkan T301 hingga T500 diisi nol.

Tabel 13. Representasi *task_mask*

Task_ID	Ketersediaan
T1	1
...	...
T300	0
T301	0
...	...
T500	0

Ruang aksi direpresentasikan sebagai vektor multi diskrit $a = [a_t, a_i, a_j]$, yang mendefinisikan keputusan penugasan atau pemindahan tugas. Formulasi ruang aksi adalah:

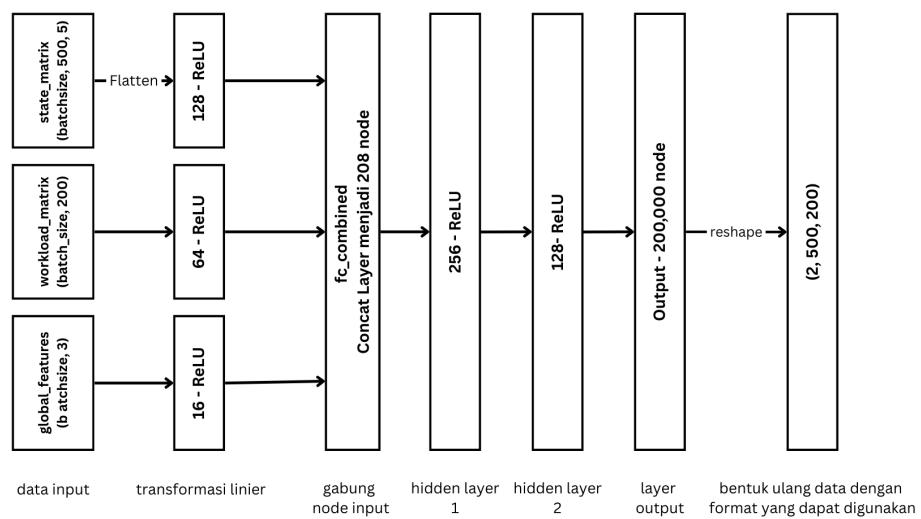
$a_t \in \{0, 1\}$ Tipe aksi, dengan 0 untuk menugaskan tugas baru dan 1 untuk memindahkan tugas ke karyawan lain.

$a_i \in \{0, \dots, 499\}$ Indeks tugas yang dipilih, hanya 0-299 aktif.

$a_j \in \{0, \dots, 199\}$ Indeks karyawan yang dipilih, hanya 0-108 aktif

3.8.3 Policy Network & Target Network

Gambar dibawah menjelaskan arsitektur *policy network* DQN yang akan diimplementasi, yang merupakan pembuat keputusan aksi yang akan dipilih pada saat optimasi. Dimulai dengan ketiga komponen *state* yang akan digunakan sebagai input pertama akan dilakukan proses transformasi linier supaya dapat disatukan menjadi input yang sama pada *neural network*. Node input yang akan diterima neural network senilai 208 node yang kemudian akan diproses pada 2 lapisan *hidden layer* sebelum memberikan output 200,000 node. Output ini terdiri dari nilai Q-Value untuk 500 tugas x 200 karyawan x 2 kemungkinan penugasan (1 merepresentasikan bahwa tugas dilakukan penugasan ulang, sedangkan 0 merepresentasikan apabila tugas tersebut adalah penugasan baru). Output *Q-values* ini kemudian ini kemudian



Gambar 11. Representasi Model DQN

3.8.3 Experience Replay

Experience replay adalah sebuah matriks yang akan digunakan untuk menyimpan pengalaman pelatihan DQN selama pembelajaran. Pada penelitian ini, ukuran *experience replay* adalah 50,000 yang berarti bahwa

experience replay dapat menampung hingga 50,000 jumlah tuple transisi yang terdiri dari (state diambil, aksi diambil, reward didapat, state setelahnya, dan parameter state terminal). Dengan kapasitas 50,000, artinya setelah 50,000 langkah, isi dari *experience replay* akan ditukar dengan transisi baru dengan proses FIFO. Artinya step awal akan dihilangkan untuk membuat ruang kepada transisi baru.



Gambar 12. Representasi *Experience Replay*

3.8.3 Fungsi Reward dan Batasan

Fungsi reward dan batasan memainkan peran kunci dalam mengarahkan DQN menuju *multi-objective optimization*, dengan mempertimbangkan kecocokan keterampilan, keseimbangan beban kerja, dan minimisasi karyawan idle. Reward dihitung dengan menggabungkan tiga komponen yaitu kecocokan berbasis WED, perubahan distribusi beban kerja, dan alokasi karyawan idle, dengan tambahan reward terminal untuk mengevaluasi performa keseluruhan. Nilai reward total dihitung sebagai $R = R_s + R_w + R_i$.

Fungsi reward pertama adalah fungsi reward untuk kecocokan keterampilan. Fungsi reward ini dilakukan tergantung dengan jenis pemberian tugas. Nilai $R_s = s_{ai,aj}$ jika $a_t = 0$, namun bernilai -0.01 jika $a_t = 1$, dan -1.0 jika aksi tidak valid. Berikut adalah formulasi perhitungan reward tersebut.

$$R_s = R_s + 50 \cdot (2 \cdot \bar{s} - 1), \text{ dengan } \bar{s} = \frac{1}{300} \sum_{i=1}^{300} s_{i, assignments[i]} \quad (31)$$

dimana:

R_s	Reward untuk kecocokan keterampilan.
$s_{ai,aj}$	Similarity score untuk tugas a_j dan karyawan a_t .
a_t	Tipe aksi (0 = assign, 1 = reassign).
\bar{s}	Rata-rata skor kecocokan untuk semua penugasan.

assignments[i] Indeks karyawan yang ditugaskan ke tugas i.

Fungsi reward kedua adalah fungsi reward untuk persebaran beban kerja. Fungsi reward ini dilakukan tergantung dengan jenis pemberian tugas. $R_w = 0.5$ jika $\sigma(W') < \sigma(W)$, nilai $R_w = -0.3$ jika $\sigma(W') > \sigma(W)$ dan 0 jika sama. Formulasi rumus tersebut adalah sebagai berikut.

$$R_w = 100 \cdot (1 - 2 \cdot \hat{\sigma}), \text{ dengan } \hat{\sigma} = \frac{\sigma(W')}{\max(\sigma(W'))} \dots \dots \dots \quad (32)$$

dimana

R_w	Reward untuk keseimbangan beban kerja.
$\sigma(W)$	Standar deviasi beban kerja sebelum aksi.
$\sigma(W')$	Standar deviasi beban kerja setelah aksi.
$\hat{\sigma}$	Standar deviasi ternormalisasi

Fungsi reward ketiga adalah fungsi reward untuk pemanfaatan karyawan. Fungsi reward ini dilakukan tergantung dengan jenis pemberian tugas. $R_i = 0.1$ jika $W_{aj} = 0$ sebelum aksi dan 0 sebaliknya. Formulasi rumus tersebut adalah sebagai berikut.

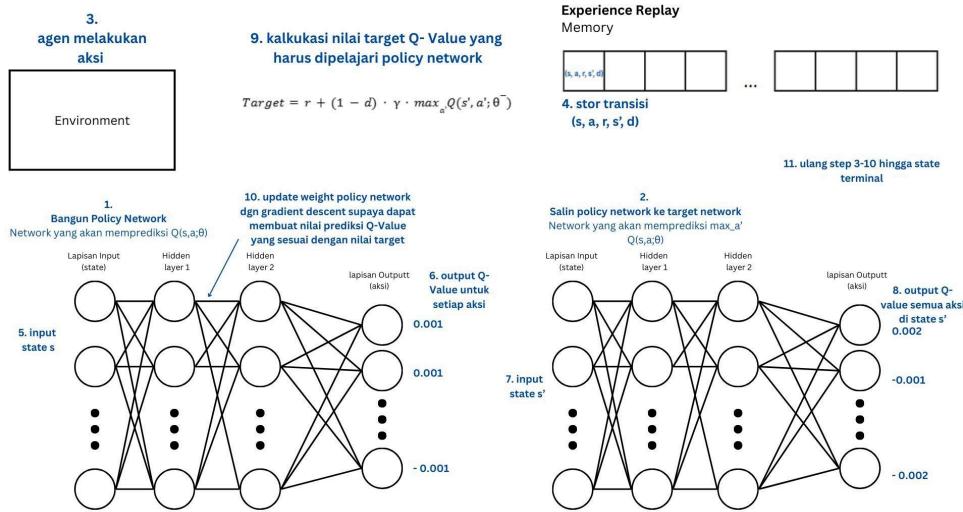
$$R_i \leftarrow R_i + 50 \cdot (1 - 2 \cdot \frac{\sum_{j=1}^{109} 1(W_j = 0)}{109}) \dots \quad (33)$$

dimana

R_i	Reward untuk minimalisir karyawan menganggur.
W_{a_j}	Beban kerja karyawan a_j sebelum aksi.
$1(w_j = 0)$	Fungsi Indikator karyawan j menganggur.

3.8.3 Proses Pelatihan

Berikut adalah representasi grafis proses pelatihan DQN. Pertama, dengan model yang dijelaskan sebelumnya, bangun policy network yang akan digunakan untuk memprediksi nilai $Q(s,a;\theta)$. Berikutnya, salin arsitektur, weight, dan bias policy network ke target network, yang nanti akan digunakan untuk memprediksi nilai $\max_{a'} Q(s', a'; \theta^-)$. Ketiga, agen akan melakukan aksi. Misal parameter angka random < nilai epsilon, maka lakukan aksi random. Sebaliknya, maka ambil aksi berdasarkan nilai Q tertinggi dari *policy network*. Selanjutnya, transisi dalam bentuk tuple (s, a, r, s', d) akan disetor di *experience replay*. Proses ini akan berlanjut hingga nilai batch *experience replay* sudah terpenuhi.



Gambar 13. Proses Pelatihan DQN

Setelah nilai transisi pada experience replay sudah lebih besar atau sama dengan jumlah batch, maka untuk setiap langkah akan dilakukan pelatihan untuk *policy network* dan *target network*. Langkah kelima adalah untuk menginput nilai s ke *policy network*, dan pada langkah enam akan dilihat nilai Q yang *random* di awal. Langkah tujuh, input state s' ke *target network* dan prediksi nilai $\max_{a'} Q(s', a'; \theta^-)$. Dapat dilihat pada langkah 8 contoh nilai Q -value untuk state s' . Langkah kesembilan adalah untuk mengambil nilai terbesar dari *target network* dan kalkulasi nilai target Q -value yang harus dipelajari *policy network*. dengan rumus dibawah.

$$Target = r + (1 - d) \cdot \gamma \cdot \max_{a'} Q(s', a'; \theta^-)$$

Dimana r adalah reward, d adalah parameter state terminal, γ adalah discount factor, dan $\max_{a'} Q(s', a'; \theta^-)$ adalah nilai maksimum Q untuk state s' dengan mengambil aksi a' pada network θ^- . Konsep ini sangat mirip dengan konsep *supervised learning*, yang dimana pada kasus

ini, fitur dataset berupa input state, dan label dataset adalah nilai target Q value yang dihasilkan target network.

Langkah kesepuluh adalah bahwa nilai target tersebut akan dihitung dengan fungsi loss dan dilakukan operasi gradient descent supaya *weight* dan *bias policy network* dapat dioptimasi sehingga dapat menghasilkan nilai target Q-value tersebut. Kemudian ulang langkah 3 hingga 10 hingga state terminal atau episode habis. Lalu ulangi kembali proses tersebut untuk setiap episode, namun pada episode selanjutnya, *weight* dan *bias* dilanjutkan berdasarkan proses pembelajaran episode sebelumnya.

Target network hanya akan diupdate weight dan bias dengan cara menyalinkan weight dan bias policy network setiap 1000 langkah, yang ditentukan pada hyperparameter TARGET_UPDATE_FREQ.

3.9 Desain Evaluasi Sistem / Metode

Akan dilakukan evaluasi performa tiga pendekatan dalam menyelesaikan masalah penugasan tugas, yaitu pendekatan *Greedy*, *Multi-Objective Optimization* (MOO) dengan *Mixed-Integer Linear Programming* (MILP), dan *Reinforcement Learning* (RL) menggunakan *Deep Q-Network* (DQN). Evaluasi performa akan dilakukan dengan fokus pada tiga metrik utama, yaitu kecocokan keterampilan yang diukur dengan *Weighted Euclidean Distance*, keseimbangan beban kerja yang diukur dengan standar deviasi beban kerja, dan jumlah karyawan menganggur.

Pendekatan *Greedy* digunakan sebagai pembanding dasar yang sederhana, di mana tugas dialokasikan ke karyawan dengan kecocokan keterampilan tertinggi tanpa memperhatikan batasan bahwa tugas harus berasal dari satu proyek yang sama, sehingga sering kali menghasilkan solusi yang tidak optimal. *Greedy* dijalankan pada CPU Intel Xeon 1 Core, yang cukup untuk algoritma sederhana ini.

Pendekatan MOO, yang menjadi acuan utama, menggunakan *solver* Gurobi untuk mengoptimalkan tiga objektif secara bersamaan, yaitu memaksimalkan kecocokan keterampilan, minimalkan variansi beban kerja, dan meminimalkan karyawan menganggur dengan mempertimbangkan semua batasan permasalahan. Karena Gurobi tidak mendukung optimasi GPU, MOO dijalankan pada mesin virtual Azure dengan spesifikasi 128 GB RAM dan 32 core CPU untuk menangani kompleksitas komputasi.

Pendekatan RL menggunakan DQN, seperti dijelaskan pada Bab 3.8, dilatih selama 50 episode menggunakan T4 GPU, CPU Intel Xeon 1 core, dan RAM 16 GB yang disewa melalui Google Colab, memanfaatkan kemampuan GPU untuk mempercepat pelatihan model neural network. Karena keterbatasan sumber daya finansial penulis, pendekatan RL hanya dapat dijalankan selama 50 episode dan untuk penelitian ini tidak dapat dilakukan optimasi *hyperparameter*.

Evaluasi dilakukan dengan membandingkan performa optimasi ketiga pendekatan berdasarkan distribusi *Weighted Euclidean Distance*, divisualisasikan melalui boxplot untuk menunjukkan kecocokan keterampilan. Beban kerja divisualisasikan melalui bar chart yang menampilkan distribusi beban kerja karyawan. Jumlah karyawan menganggur juga divisualisasikan dengan bar chart untuk membandingkan efisiensi alokasi. Setelah itu, dengan model yang sudah ada, akan dilakukan inferensi dengan dataset berbeda dengan yang digunakan untuk pelatihan.

3.10 Pengembangan Sistem

3.10.1 Analisis Kebutuhan Sistem

Pengembangan sistem bertujuan untuk mendukung proses penugasan tugas yang optimal dengan memanfaatkan pendekatan *Reinforcement Learning* berbasis *Deep Q-Network* (DQN), yang mengoptimalkan kecocokan keterampilan, keseimbangan beban kerja, dan pemanfaatan karyawan. Sistem ini dirancang untuk memudahkan admin dalam mengelola data tugas dan karyawan, menjalankan model DQN, serta

menganalisis hasil penugasan melalui antarmuka yang intuitif. Berikut adalah kebutuhan fungsional dan non-fungsional yang diidentifikasi untuk sistem ini.

Kebutuhan fungsional adalah kebutuhan yang berdasarkan proses yang mampu disediakan oleh sistem dan mencangkup kebutuhan dasar pengguna tersebut berupa fitur, layanan dan fungsi. Adapun kebutuhan fungsional sistem yang akan dirancang sebagai berikut.

1. Autentikasi Pengguna: Sistem harus menyediakan fitur login untuk admin menggunakan email dan kata sandi, dengan data autentikasi disimpan dalam basis data SQLite. Fitur pendaftaran (*signup*) tidak diperlukan untuk pengguna umum, sehingga registrasi hanya dapat dilakukan oleh admin secara manual untuk menjaga keamanan akses.
2. Pengunggahan Data: Sistem harus memungkinkan admin untuk mengunggah dataset tugas dan karyawan dalam format CSV, dengan batasan maksimal 500 tugas dan 200 karyawan untuk menjaga performa. Dataset harus mencakup informasi seperti ID tugas, ID proyek, story points, dan kategori keterampilan untuk tugas, serta ID karyawan, peran, dan keterampilan untuk karyawan. Sistem juga harus memvalidasi kesesuaian format data, seperti memastikan kolom keterampilan pada dataset tugas dan karyawan memiliki dimensi yang sama.
3. Manajemen Basis Data: Sistem harus mampu menyimpan data tugas dan karyawan ke dalam basis data SQLite, termasuk informasi keterampilan yang relevan, dan mengelola data penugasan (*assignments*) yang dihasilkan oleh model DQN, seperti ID tugas, ID karyawan, skor kecocokan, dan status penggerjaan. Sistem juga harus dapat menghapus data lama pengguna sebelum mengunggah data baru untuk menjaga integritas data.

4. Penugasan Tugas dengan Model DQN: Sistem harus mengintegrasikan model DQN yang telah dilatih untuk menjalankan proses penugasan tugas secara otomatis setelah data diunggah, menghasilkan alokasi tugas berdasarkan kecocokan keterampilan, dengan hasil disimpan dalam basis data dan dapat diakses untuk analisis lebih lanjut.
5. Project Backlog: Sistem harus menampilkan daftar tugas dalam bentuk tabel interaktif yang mencakup informasi seperti nama tugas, nama proyek, story points, karyawan yang ditugaskan (PIC), dan status penggerjaan (*To Do*, *In Progress*, *Done*). Admin harus dapat mengedit data ini, seperti mengubah PIC atau status tugas, untuk mendukung pengelolaan proyek secara manual jika diperlukan.
6. Visualisasi Performa: Sistem harus menyediakan visualisasi metrik performa penugasan, termasuk boxplot distribusi skor kecocokan keterampilan, bar chart total story points per karyawan, dan bar chart jumlah tugas per karyawan, untuk membantu admin menganalisis hasil penugasan dari model DQN.
7. Ekspor Data: Sistem harus menyediakan fitur untuk mengunduh hasil penugasan dalam format CSV, memungkinkan admin menyimpan dan menganalisis data di luar sistem.

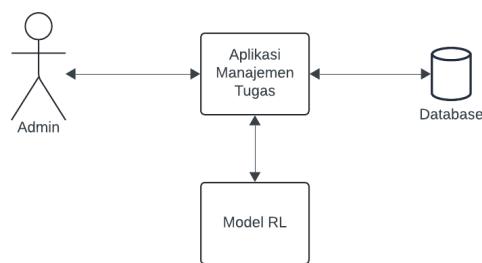
Kebutuhan non fungsional merupakan sekumpulan batasan, karakteristik, dan properti pada sistem, baik dalam pengembangan maupun operasional. Kebutuhan non fungsional sistem yang akan diimplementasi adalah sebagai berikut.

1. Performa: Sistem harus mampu memproses dan menyimpan data hingga 500 tugas dan 200 karyawan dengan waktu respons yang cepat, menggunakan basis data SQLite untuk efisiensi penyimpanan. Proses evaluasi model DQN harus berjalan efisien

- tanpa menyebabkan penundaan yang signifikan pada antarmuka pengguna.
2. Skalabilitas: Sistem harus dirancang untuk mendukung jumlah tugas dan karyawan yang bervariasi, dengan batasan maksimal yang telah ditentukan, dan mampu menangani perubahan kebutuhan data tanpa memerlukan perubahan signifikan pada struktur sistem.
 3. Kemudahan Penggunaan: Antarmuka sistem harus intuitif, dengan navigasi yang jelas antara halaman login, upload data, project backlog, dan dashboard. Visualisasi harus mudah dipahami, dan fitur seperti upload file seta tabel interaktif harus mendukung pengelolaan data yang efisien oleh admin.

3.10.2 Desain Sistem

Secara keseluruhan, sistem dapat dilihat sesuai dengan gambar 10. Pengguna akan mengakses aplikasi manajemen tugas dan dapat mengunggah atau mengambil data dari database. Jika pengguna ingin melakukan penugasan terautomasi, maka permintaan tersebut akan dikirimkan ke model RL yang akan memproses penugasan tersebut.

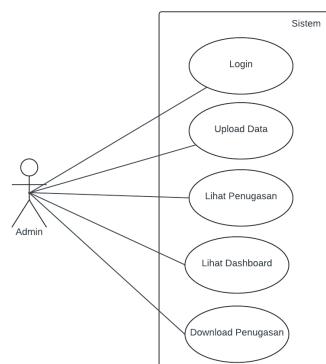


Gambar 14. Arsitektur Keseluruhan Sistem

Use Case Diagram menggambarkan interaksi antara Admin sebagai aktor utama dengan sistem. Diagram ini menunjukkan lima use case utama yang berada dalam batas sistem, yaitu login, upload data, lihat penugasan, lihat dashboard, download penugasan.

Use case "Login" memungkinkan admin untuk mengakses sistem dengan memasukkan email dan kata sandi. Setelah login, admin dapat menggunakan "Upload Data" untuk mengunggah file CSV yang berisi data tugas dan karyawan, sebagai langkah awal sebelum penugasan dilakukan. Use case "Lihat Penugasan" memungkinkan admin untuk melihat dan mengedit hasil penugasan dalam bentuk tabel interaktif, yang mencakup informasi seperti nama tugas, proyek, story points, dan karyawan yang ditugaskan.

Use case "Lihat Dashboard" menyediakan visualisasi performa penugasan, seperti distribusi beban kerja dan skor kecocokan keterampilan, untuk membantu admin dalam analisis. Terakhir, "Download Penugasan" memungkinkan admin untuk mengunduh hasil penugasan dalam format CSV untuk keperluan lebih lanjut. Semua use case ini dihubungkan langsung dengan admin melalui garis asosiasi, menunjukkan bahwa admin memiliki akses penuh terhadap semua fungsi sistem.



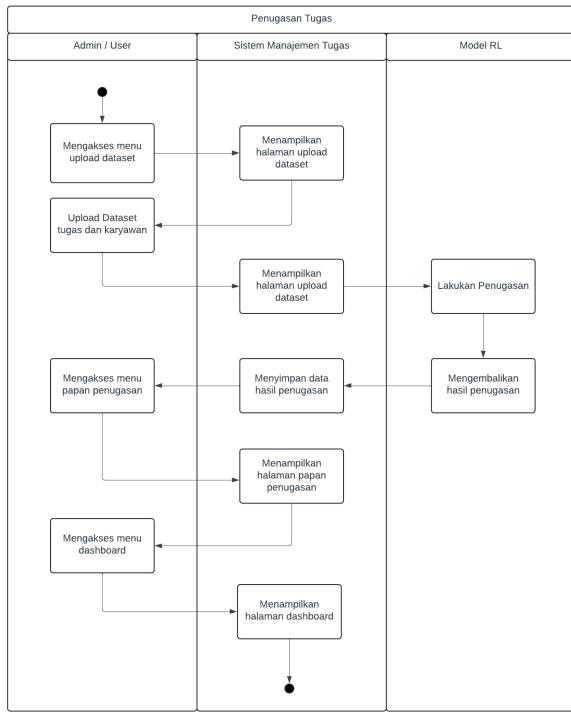
Gambar 15. Use Case Diagram

Activity Diagram menggambarkan alur kerja sistem dalam proses penugasan tugas, dengan tiga swimlanes yang memisahkan tindakan berdasarkan aktor atau komponen: Admin, Sistem Manajemen Tugas, dan Model RL. Alur dimulai dari Admin yang mengakses menu upload dataset untuk mengunggah file CSV tugas dan karyawan. Sistem Manajemen

Tugas kemudian menampilkan halaman upload dataset, memungkinkan admin untuk mengunggah data. Setelah data diunggah, sistem menyimpan data ke dalam basis data dan melanjutkan dengan menjalankan Model RL untuk melakukan penugasan tugas menggunakan algoritma DQN.

Model RL menghasilkan hasil penugasan, yang kemudian disimpan kembali ke basis data oleh sistem. Selanjutnya, Admin dapat mengakses menu papan penugasan untuk melihat hasil penugasan dalam bentuk tabel interaktif, yang mencakup informasi seperti nama tugas, proyek, story points, karyawan yang ditugaskan, dan status penggerjaan. Admin juga dapat mengakses menu dashboard untuk melihat visualisasi performa penugasan, seperti distribusi beban kerja, skor kecocokan keterampilan, dan pemanfaatan karyawan.

Terakhir, sistem menampilkan halaman dashboard dengan visualisasi tersebut, dan admin dapat mengunduh hasil penugasan dalam format CSV jika diperlukan. Alur ini menunjukkan bagaimana sistem mengintegrasikan interaksi admin, pengelolaan data, dan model RL untuk menghasilkan penugasan tugas yang optimal.



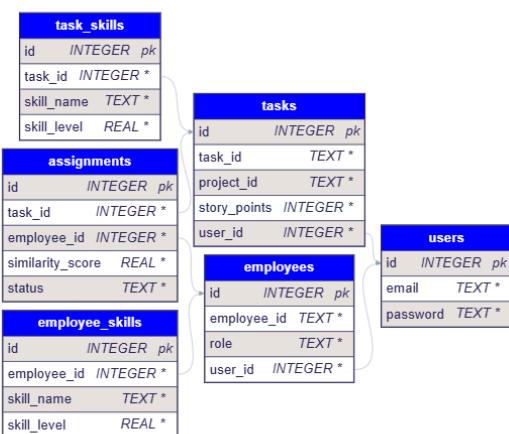
Gambar 16. Activity Diagram

ERD menggambarkan struktur basis data sistem dengan enam entitas, yang akan dijelaskan sebagai berikut.

- Entitas users memiliki atribut id (primary key), email, dan password, yang semuanya wajib diisi (ditandai dengan tanda bintang).
- Entitas tasks memiliki atribut id (primary key), task_id, project_id, story_points, dan user_id (foreign key ke users), dengan task_id, project_id, story_points, dan user_id sebagai kolom wajib.
- Entitas employees memiliki atribut id (primary key), employee_id, role, dan user_id (foreign key ke users), dengan employee_id, role, dan user_id wajib diisi.
- Entitas task_skills memiliki atribut id (primary key), task_id (foreign key ke tasks), skill_name, dan skill_level, dengan semua kolom wajib.

- Entitas `employee_skills` memiliki atribut `id` (primary key), `employee_id` (foreign key ke `employees`), `skill_name`, dan `skill_level`, juga dengan semua kolom wajib.
- Entitas `assignments` memiliki atribut `id` (primary key), `task_id` (foreign key ke `tasks`), `employee_id` (foreign key ke `employees`), `similarity_score`, dan `status`, dengan semua kolom wajib.

Relasi antar entitas ditunjukkan dengan garis: `users` memiliki banyak `tasks` dan `employees` (*one-to-many*), `tasks` memiliki banyak `task_skills` (*one-to-many*), `employees` memiliki banyak `employee_skills` (*one-to-many*), dan `assignments` menghubungkan `tasks` dan `employees` (*one-to-one dengan tasks, many-to-one dengan employees*).



Gambar 17. Entity Relationship Diagram

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Pengumpulan data untuk penelitian ini telah berhasil dilakukan sesuai rencana yang diuraikan pada Bab 3.3, menghasilkan dua dataset utama dari Direktorat Digital Business PT. Telkom Indonesia STO Kebayoran Baru, yaitu dataset karyawan dan dataset tugas. Dataset karyawan mencakup data dari 109 karyawan, masing-masing dengan informasi ID karyawan, peran, dan skor untuk 65 metrik kompetensi (skala 0–5). Dataset tugas terdiri dari 300 tugas yang mewakili lima proyek (P1–P5), dengan fitur ID tugas, ID proyek, story point (1–20), dan kebutuhan kompetensi yang selaras dengan dataset karyawan.

Dataset dikumpulkan dengan metode survei mandiri yang telah sebelumnya dilakukan Direktorat Digital Business PT. Telkom Indonesia STO Kebayoran Baru. Hasil pengumpulan data ini menjadi dasar untuk analisis eksplorasi data, termasuk pemeriksaan kelayakan dataset, distribusi story point, dan struktur proyek, yang dibahas pada sub bab berikutnya.

4.2 Eksplorasi Data

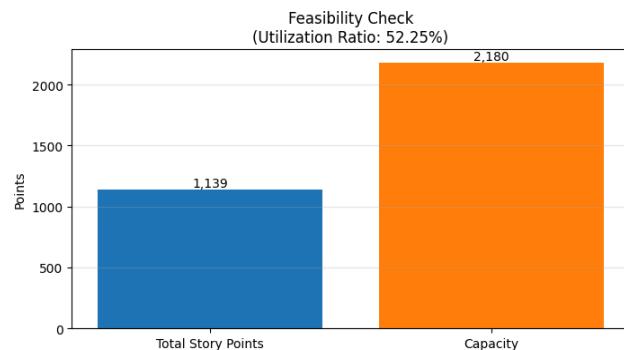
4.2.1 Pemeriksaan Kelayakan Dataset (*Feasibility Check*)

Analisis pemeriksaan kelayakan dilakukan untuk mengevaluasi keseimbangan antara total story points dan kapasitas tim dalam konteks penugasan tugas. Berdasarkan data yang diperoleh, total story points dari 300 tugas adalah 1139, sementara kapasitas tim, yang dihitung sebagai 109 karyawan dengan masing-masing kapasitas maksimum 20 story points, mencapai 2180. Rasio pemanfaatan, yang didefinisikan sebagai perbandingan total story points terhadap kapasitas, adalah 0.52 atau 52%.

Visualisasi berupa diagram batang menunjukkan perbandingan ini dengan jelas, di mana kapasitas tim jauh melebihi total story points,

mengindikasikan adanya ruang kapasitas yang signifikan. Rasio utilisasi ini memberikan gambaran awal tentang fleksibilitas dalam alokasi sumber daya untuk penugasan tugas. Implikasi teknis dari temuan ini bagi model optimasi adalah bahwa ruang solusi menjadi cukup longgar, memungkinkan model untuk menemukan solusi yang feasible dengan relatif mudah. Namun, kapasitas berlebih ini dapat menyebabkan model cenderung menghasilkan solusi yang suboptimal jika tidak ada batasan tambahan, seperti distribusi beban kerja yang merata atau prioritas pada efisiensi alokasi.

Oleh karena itu, model perlu dirancang dengan fungsi objektif yang mendorong pemerataan beban kerja atau simulasi dengan beban kerja tambahan untuk menguji batas kapasitas. Dari perspektif bisnis, rasio pemanfaatan 52% menunjukkan potensi *overstaffing* atau kurangnya tugas yang dialokasikan, sehingga organisasi dapat mempertimbangkan untuk menambah proyek baru atau menyesuaikan jumlah karyawan guna meningkatkan efisiensi biaya.



Gambar 18. Hasil Pemeriksaan Kelayakan Dataset

4.2.2 Distribusi *Story Point*

Distribusi story points dianalisis untuk memahami tingkat kompleksitas tugas dalam dataset menggunakan deret Fibonacci.

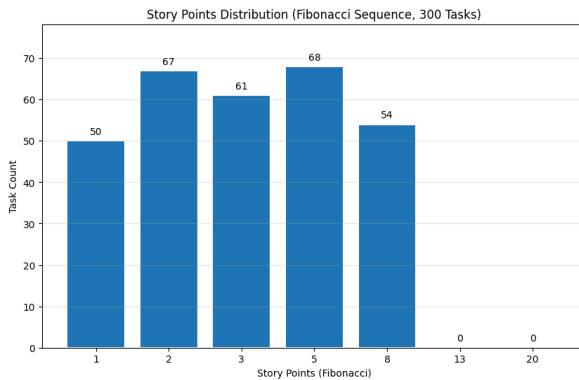
Berdasarkan data, distribusi story points dari 300 tugas adalah sebagai berikut:

- 50 tugas dengan *story point* 1,
- 67 tugas dengan *story point* 2,
- 61 tugas dengan *story point* 3,
- 68 tugas dengan *story point* 5,
- 54 tugas dengan *story point* 8,
- dan tidak ada tugas dengan *story point* 13 atau 20.

Visualisasi bar chart menunjukkan bahwa mayoritas tugas (246 tugas) memiliki story points antara 1 hingga 5, dengan puncak pada story point 5, sementara hanya 54 tugas yang memiliki story points antara 5 hingga 10, dan tidak ada tugas dengan story points lebih dari 10. Pola ini mengindikasikan bahwa tugas dalam proyek cenderung memiliki kompleksitas rendah hingga sedang.

Implikasi teknis dari distribusi ini bagi model adalah bahwa algoritma penugasan perlu dioptimalkan untuk menangani volume tugas kecil yang tinggi, yang memerlukan strategi alokasi granular. Ketidakhadiran tugas dengan story points besar (13 dan 20) menyederhanakan model karena tidak perlu menangani distribusi beban kerja ekstrem, tetapi juga menunjukkan bahwa model belum diuji untuk skenario tugas berat, sehingga simulasi tambahan mungkin diperlukan untuk memastikan robustitas. Objective function dapat difokuskan pada efisiensi penyebaran tugas kecil untuk menghindari bottleneck pada karyawan tertentu.

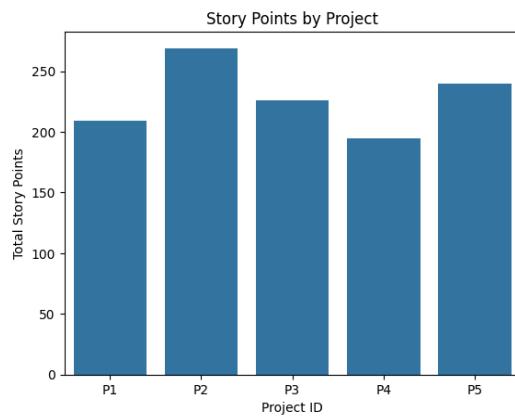
Dari perspektif bisnis, dominasi tugas kecil mencerminkan pendekatan proyek yang inkremental, yang sesuai dengan metodologi *agile*, tetapi dapat meningkatkan risiko *context switching* bagi karyawan, sehingga memerlukan manajemen tugas yang lebih terstruktur.



Gambar 19. Hasil Distribusi *Story Point* Dataset

4.2.3 Analisis Struktur Proyek

Analisis struktur proyek bertujuan untuk memahami distribusi beban kerja dan kebutuhan keterampilan di lima proyek (P1 hingga P5). Data statistik proyek menunjukkan bahwa P1 memiliki 209 story points dengan 56 tugas, P2 memiliki 269 story points dengan 66 tugas, P3 memiliki 226 story points dengan 61 tugas, P4 memiliki 195 story points dengan 52 tugas, dan P5 memiliki 240 story points dengan 65 tugas.



Gambar 20. Jumlah *Story Point* Setiap Proyek

Implikasi teknis bagi model adalah bahwa algoritma penugasan harus mampu menangani distribusi karyawan secara proporsional untuk menghindari ketimpangan antar proyek, terutama dengan P2 yang

memiliki beban kerja terbesar. Kebutuhan keterampilan yang bervariasi menyarankan penggunaan constraint spesifik per proyek atau objective function yang mempertimbangkan profil keterampilan unik, misalnya dengan memprioritaskan karyawan dengan skor statistik di atas 3.0 untuk P2.

Dari perspektif bisnis, proyek seperti P2 yang memiliki beban kerja besar mungkin memerlukan alokasi sumber daya tambahan, sementara P4 dapat diprioritaskan lebih rendah, memungkinkan organisasi untuk menyeimbangkan fokus antar proyek.

4.2 Hasil Pra-pemrosesan Data

Kedua pendekatan melakukan pra-pemrosesan dasar yang sama untuk memastikan konsistensi data input. Pertama, seleksi fitur. Proses ini hanya mempertahankan kolom yang relevan yaitu 65 kolom skill, story_points, dan project_id. Kolom lain seperti 'no' dan 'Role' diabaikan karena tidak berkontribusi dalam analisis. Berikutnya adalah normalisasi nilai skill yang awalnya dalam skala 1-5 dinormalisasi ke range [0,1] dengan dibagi dengan 5. Ini dilakukan untuk memastikan semua fitur memiliki skala yang sama untuk pendekatan RL karena mencegah dominasi fitur dengan skala besar.

Berikutnya adalah penanganan missing value. Nilai yang hilang (NaN) diimputasi dengan 0, mengasumsikan bahwa ketiadaan nilai berarti tidak memiliki skill tersebut. Data diorganisir dalam bentuk dataframe Pandas dengan tabel karyawan dan tabel tugas. Bentuk yang akan diproses semua pendekatan nantinya seperti berikut.

Tabel 14. Fitur dataset *Employee* setelah pra-pemrosesan awal

Employee_id	Mathematics.LinearAlgebra	...	MLOPS.Model Deployment
Employee_1	0.6		0.2

Tabel 15. Fitur dataset *Task* setelah pra-pemrosesan awal

Task_id	Project_id	Story Point	Mathematics.LinearAlgebra	...	MLOPS.Model
---------	------------	-------------	---------------------------	-----	-------------

					Deployment
Task_1	P1	0.15	0.8		1.0

Terakhir, perhitungan WED dilakukan di awal untuk setiap pasangan karyawan-tugas untuk menghindari perhitungan berulang kali saat pelatihan. Berikut adalah representasi matriks WED untuk tahap pra pemrosesan umum.

Tabel 16. Representasi matriks WED awal

Task_ID	Talent 1	...	Talent 109
T1	2.35	...	5.20
...
T109	2.75	...	5.80

4.2.1 Pra-pemrosesan Khusus Pendekatan *Greedy*

Pendekatan *greedy* memerlukan beberapa tahap pra-pemrosesan khusus untuk mendukung mekanisme penugasannya yang bersifat sekuensial. Pertama, dilakukan permutasi acak urutan tugas untuk memastikan proses penugasan tidak terpengaruh oleh pola tertentu dalam urutan data asli. Teknik ini penting untuk menghindari bias sistemik dimana tugas-tugas tertentu selalu dipertimbangkan lebih dahulu.

Kedua, disiapkan struktur data khusus berupa *dictionary* penugasan untuk mencatat penugasan per karyawan, dan array workload berukuran 109 (sesuai jumlah karyawan) untuk memantau akumulasi beban kerja. Struktur ini diinisialisasi dengan nilai default sebelum iterasi penugasan dimulai. Berikut adalah hasil pra pemrosesan:

Tabel 17. Hasil Pra Pemrosesan untuk Pendekatan *Greedy*

Komponen	Representasi

Keterampilan Karyawan	Matriks (109 x 65), nilai keterampilan dalam skala [0, 1]
Keterampilan Tugas	Matriks (300 x 65), nilai keterampilan dalam skala [0, 1]
Urutan Tugas Acak	Array indeks tugas, ukuran 300
Struktur Penugasan	Dictionary dengan 109 kunci dan array ukuran 109

4.2.2 Pra-pemrosesan Khusus Pendekatan *MOO*

Pra pemrosesan untuk pendekatan *Multi-Objective Optimization* (MOO) dirancang untuk mendukung optimasi tiga objektif: meminimalkan karyawan idle, memaksimalkan kecocokan keterampilan, dan menyeimbangkan beban kerja. Langkah-langkah ini menghasilkan struktur data untuk model Gurobi. Berikut adalah hasil pra pemrosesan MOO, disajikan dalam tabel. Tabel 3. Struktur Data Pendekatan *MOO*

Tabel 18. Hasil Pra Pemrosesan untuk Pendekatan MOO

Komponen	Representasi
Data Karyawan	Matriks (109 x 65), nilai keterampilan dalam skala 0–5
Data Tugas	Matriks (300 x 65), nilai keterampilan dalam skala 0–5
Pengelompokan Tugas per Proyek	Dictionary dengan 5 kunci (P1–P5)
Story Points	Dictionary dengan 300 pasangan

Project ID	Dictionary dengan 300 pasangan
Similarity Score	Matriks (109 \times 300), nilai similarity berbasis WED
Data untuk Model	Struktur indeks untuk variabel keputusan Gurobi

4.2.3 Pra-pemrosesan Khusus Pendekatan *RL*

Pra Pemrosesan untuk pendekatan *Deep Q-Network* (DQN) bertujuan menghasilkan struktur data berbasis tensor untuk lingkungan *Reinforcement Learning* (RL) dalam TaskAssignmentEnv. Langkah-langkah ini mendukung pelatihan model DQN menggunakan PyTorch. Berikut adalah hasil pra pemrosesan DQN, disajikan dalam tabel.

Tabel 19. Hasil Pra Pemrosesan untuk Pendekatan RL

Komponen	Representasi
Keterampilan Karyawan	Tensor (200 x 65), nilai keterampilan dalam skala [0, 1]
Keterampilan Tugas	Tensor (500 x 65), nilai keterampilan dalam skala [0, 1]
Project ID	Tensor ukuran 500, indeks numerik proyek
Story Points	Tensor ukuran 500, nilai beban kerja tugas
State Matrix	Tensor (500 x 5), dengan nilai WED tertinggi top 5

4.3 Uji Coba Pendekatan *Greedy*

4.3.1 Implementasi Pendekatan *Greedy*

Implementasi pendekatan greedy untuk penugasan tugas dilakukan melalui beberapa tahapan utama. Pertama, data karyawan dan tugas dimuat dari dataset yang telah dipersiapkan, dengan melakukan normalisasi nilai skill ke dalam range [0,1] untuk memastikan konsistensi perhitungan. Matriks keterampilan karyawan (109×65) dan kebutuhan skill tugas (300×65) menjadi dasar perhitungan kompatibilitas.

Proses penugasan mengadopsi mekanisme greedy dengan mempertimbangkan dua faktor utama: kapasitas beban kerja maksimum dan nilai WED. Batasan satu karyawan hanya dapat mengambil pekerjaan dari proyek yang sama diabaikan pada pendekatan ini. Implementasi ini memastikan preferensi diberikan kepada karyawan yang memiliki keterampilan paling mendekati kebutuhan tugas.

Mekanisme penugasan dilakukan secara iteratif dengan memproses tugas dalam urutan acak untuk menghindari bias. Untuk setiap tugas, algoritma mencari karyawan tersedia dengan skor WED terkecil yang masih memiliki kapasitas cukup. Hasil penugasan disimpan dalam struktur data yang mencatat informasi karyawan, proyek, tugas yang dialokasikan, total story points, serta similarity score masing-masing penugasan.

Setelah proses penugasan selesai, dilakukan standarisasi skor WED terhadap kasus terburuk dan terbaik untuk memungkinkan analisis komparatif. Implementasi juga menghitung metrik evaluasi berupa standar deviasi beban kerja dan jumlah karyawan tidak mendapatkan tugas sebagai indikator kualitas solusi.

Pendekatan *Greedy* memberikan solusi yang cepat dan efisien secara komputasi, meskipun tidak menjamin optimalitas global, dengan kompleksitas waktu $O(n \times m)$ untuk n tugas dan m karyawan. Implementasi

menggunakan Python dengan memanfaatkan library NumPy untuk operasi matriks dan Pandas untuk manajemen data.

Tabel 20. Pseudocode Implementasi *Greedy*

Algoritma Pendekatan *Greedy*

Inisialisasi:

- Baca data task $T \in R^{(300 \times 65)}$ dengan kolom:
 - * task_id, project_id, story_points, skill_requirements[1..65]
- Baca data employee $E \in R^{(109 \times 65)}$ dengan kolom:
 - * employee_id, skills[1..65]
- Normalisasi skill values ke range [0,1]
- Inisialisasi:
 - * assignments[109] = {} # Dictionary penugasan
 - * workload[109] = 0 # Beban kerja per employee
 - * weds_raw = [] # Penyimpanan similarity scores

Fungsi calculate_wed(emp_skills, task_skills, $\alpha=0.5$):

```
weights = 1 / (1 +  $\alpha$  * max(0, emp_skills - task_skills))
wed =  $\sqrt{(\sum(weights * (emp_skills - task_skills)^2))}$ 
return wed
```

Proses Penugasan:

Untuk setiap task dalam random permutation(300):

```
task_skills = T[task].skills
sp = T[task].story_points
best_emp = None
best_wed =  $\infty$ 
```

Untuk setiap employee dalam 109 employees:

Jika $workload[emp] + sp \leq 20$:

```

curr_wed = calculate_wed(E[emp].skills, task_skills)

Jika curr_wed < best_wed:
    best_wed = curr_wed
    best_emp = emp

```

Jika best_emp ditemukan:

```

assignments[best_emp].append(task)
workload[best_emp] += sp
weds_raw.append(best_wed)

```

Standardisasi WED Scores:

```

wed_worst =  $\sqrt{(\sum((1/(1+\alpha*1)) * (0-1)^2 * 65))}$  # Kasus terburuk
wed_best = 0 # Kasus terbaik
weds_standardized = [(wed_worst - wed)/(wed_worst - wed_best)
untuk wed dalam weds_raw]

```

Penyimpanan Hasil:

Untuk setiap employee:

```

projects = [task.project_id untuk task dalam assignments[emp]]
tasks = [task.task_id untuk task dalam assignments[emp]]
sum_sp =  $\sum$ (task.story_points)
wasted_sp = 20 - sum_sp
similarity_scores = [wed untuk task dalam assignments[emp]]
Simpan ke dataframe hasil

```

Hitung Metrik:

```

std_dev = standar_deviasi(sum_sp semua employee)
idle_count = jumlah employee dengan sum_sp == 0

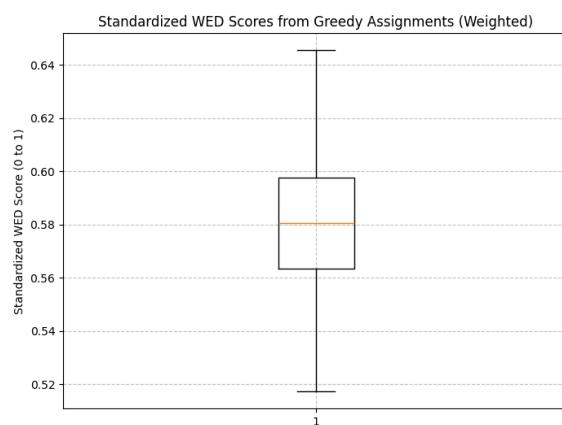
```

End Algoritma

4.3.2 Evaluasi Pendekatan *Greedy*

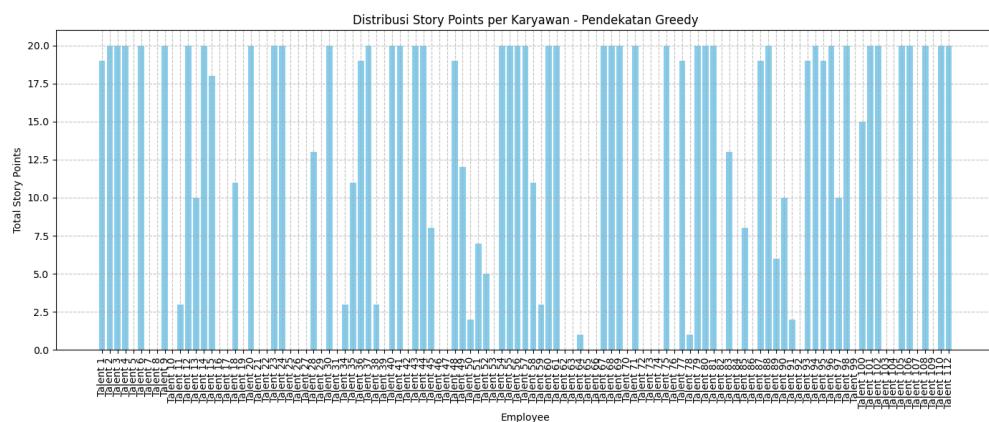
Pendekatan greedy diterapkan untuk mengevaluasi efektivitas penugasan tugas menggunakan rumus *Weighted Euclidean Distance* (WED), diikuti dengan standarisasi skor untuk analisis lebih lanjut. Pendekatan ini bertujuan untuk memberikan dasar awal dalam penugasan tugas berdasarkan kecocokan keterampilan, dengan mempertimbangkan kapasitas maksimum 20 *story points* per karyawan dan memastikan setiap tugas hanya dialokasikan kepada satu karyawan. Hasil uji coba pendekatan *greedy* memberikan wawasan penting mengenai kecocokan keterampilan dan distribusi beban kerja, yang dianalisis melalui skor WED terstandarisasi dan alokasi story points per karyawan. Perlu diketahui bahwa nilai yang dihasilkan pada bagian ini berpotensi untuk berubah sedikit pada bagian evaluasi perbandingan metode karena faktor kurang konsistensi pendekatan ini.

Skor WED terstandarisasi memiliki rata-rata 0.582, median 0.580, standar deviasi 8.93, dan 35 karyawan tidak bekerja. Walaupun nilai WED terlihat cukup baik sekilas, variansi dan jumlah karyawan yang tidak bekerja sangat tidak optimal.



Gambar 21. Boxplot Nilai WED Metode *Greedy*

Analisis distribusi beban kerja menunjukkan bahwa total *story points* per karyawan bervariasi, dengan beberapa karyawan mendapatkan alokasi mendekati kapasitas maksimum (20 story points), sementara yang lain tidak mendapatkan alokasi sama sekali (0 story points). Standar deviasi beban kerja antar karyawan adalah 8.93, yang mengindikasikan variasi yang cukup signifikan dalam distribusi beban kerja.



Gambar 22. Histogram Distribusi *Story Point* Metode *Greedy*

Standar deviasi beban kerja yang cukup tinggi (8.93) menunjukkan adanya ketidakseimbangan dalam distribusi story points antar karyawan. Beberapa karyawan mendapatkan beban kerja yang mendekati batas kapasitas, sementara yang lain tidak mendapatkan alokasi sama sekali. Hal ini menyarankan perlunya mekanisme tambahan dalam pendekatan *greedy* untuk memastikan distribusi beban kerja yang lebih merata, misalnya dengan memprioritaskan karyawan yang belum mendapatkan tugas.

Outlier pada skor WED (0.582) dapat menjadi titik analisis lebih lanjut untuk mengidentifikasi kombinasi karyawan-tugas yang menghasilkan kecocokan keterampilan lebih tinggi. Kombinasi ini dapat digunakan untuk memperbaiki heuristik penugasan dalam pendekatan *greedy*. Selain itu, pendekatan alternatif seperti simulated annealing atau reinforcement learning dapat dipertimbangkan untuk meningkatkan

eksplorasi ruang solusi, sehingga menghasilkan kecocokan keterampilan yang lebih optimal dan distribusi beban kerja yang lebih seimbang.

Dari perspektif bisnis, konsistensi skor WED mencerminkan distribusi tugas yang adil dari segi kecocokan keterampilan, yang dapat meningkatkan kepuasan karyawan dan efisiensi penyelesaian tugas. Namun, ketidakseimbangan beban kerja yang ditunjukkan oleh standar deviasi yang tinggi dapat menimbulkan risiko ketidakpuasan di kalangan karyawan yang kelebihan beban atau merasa kurang dimanfaatkan. Outlier pada skor WED dapat menjadi peluang untuk mengidentifikasi praktik terbaik dalam penugasan tugas, yang kemudian dapat direplikasi untuk meningkatkan performa tim secara keseluruhan. Pendekatan greedy ini memberikan fondasi yang kuat sebagai baseline, tetapi perbaikan lebih lanjut diperlukan untuk mengatasi keterbatasan dalam eksplorasi solusi dan pemerataan beban kerja.

4.4 Hasil MOO

Pendekatan *Multi-Objective Optimization* (MOO) dengan metode *Weighted Sum Mixed-Integer Linear Programming* diterapkan untuk mengoptimasi penugasan tugas kepada karyawan dengan mempertimbangkan tiga objektif utama, yaitu meminimalkan jumlah karyawan yang tidak mendapatkan tugas, memaksimalkan skor WED yang mencerminkan kecocokan antara keahlian karyawan dan tugas, serta meminimalisir variansi beban kerja untuk memastikan distribusi yang merata di antara karyawan. Hasil dari pendekatan ini dianalisis dan dibandingkan dengan pendekatan berbasis objektif tunggal (Objektif 1, Objektif 2, dan Objektif 3) untuk mengevaluasi efektivitas MOO dalam mencapai keseimbangan di antara ketiga objektif tersebut.

4.4.1 Implementasi MOO

Implementasi dilakukan menggunakan Gurobi dengan model *Mixed-Integer Linear Programming* (MILP). Proses melibatkan inisialisasi matriks keterampilan karyawan dan kebutuhan tugas, perhitungan skor

similarity menggunakan WED, pembuatan model dengan variabel keputusan (x untuk penugasan, y untuk alokasi proyek, max_workload untuk beban kerja maksimum), pengaturan kendala, optimasi masing-masing objektif untuk nilai ideal, dan optimasi gabungan dengan bobot 0.05 (idle employees), 0.85 (similarity error), dan 0.1 (maximum workload). Hasil penugasan disimpan dalam format CSV.

Hasil optimasi MOO menghasilkan penugasan tugas yang mencakup informasi karyawan, proyek, tugas yang dialokasikan, total story points, story points yang tidak terpakai, dan skor similarity. Hasil ini disimpan dalam format CSV untuk analisis lebih lanjut. Proses implementasi dioptimalkan dengan parameter Gurobi seperti presolve agresif, fokus pada batas terbaik, dan toleransi *optimality gap* sebesar 2.5%, untuk meningkatkan efisiensi komputasi.

Karena implementasi kode yang panjang tidak memungkinkan untuk dijabarkan pada bagian ini, akan dijelaskan dengan pseudocode. Proses ini dijelaskan pada pseudocode dibawah

Tabel 21. Pseudocode Implementasi MOO

Algoritma Multi-Objective Optimization
<p>Inisialisasi:</p> <ul style="list-style-type: none"> - Employee skills matrix $E \in R^{(m \times s)}$ - Task requirements matrix $T \in R^{(n \times s)}$ - Story points vector $S \in R^n$ - Company-task mapping $C \in \{1..k\}^n$ - Max workload capacity W_{\max}
<p>Hitung similarity score:</p> <p>For setiap task $i \in \{1..n\}$, employee $j \in \{1..m\}$:</p> <pre>common_skills ← $E[j,:] \cap T[i,:]$</pre>

Jika common_skills $\neq \emptyset$:

$\text{sim}[i,j] \leftarrow 1/(1 + \sqrt{(\sum(w_e * (E[j,s] - T[i,s])^2)}) \quad \forall s \in \text{common_skills}$

Else:

$\text{sim}[i,j] \leftarrow 0$

Bangun model optimisasi:

Inisialisasi variabel:

- $x[i,j,k] \in \{0,1\} \quad \forall i \in \text{tasks}, j \in \text{employees}, k \in \text{companies}$
- $y[j,k] \in \{0,1\} \quad \forall j \in \text{employees}, k \in \text{companies}$
- $\text{max_workload} \in [0, W_{\text{max}}]$

Constraints:

1. Setiap task ditugaskan ke 1 employee:

$$\sum(x[i,j,C[i]] \quad \forall j \in \text{employees}) = 1 \quad \forall i \in \text{tasks}$$

2. Employee di maks 1 company:

$$\sum(y[j,k] \quad \forall k \in \text{companies}) \leq 1 \quad \forall j \in \text{employees}$$

3. Relasi x dan y:

$$x[i,j,k] \leq y[j,k] \quad \forall i,j,k$$

4. Batasan workload:

$$\sum(S[i]*x[i,j,k] \quad \forall i \in \text{tasks}) \leq \text{max_workload} \quad \forall j,k$$

Optimasi single objective:

1. Minimasi idle employees:

$$\min \sum(1 - \sum(y[j,k] \quad \forall k)) \quad \forall j$$

solve → dapat solusi X1

2. Minimasi similarity error:

$$\min \sum((1-\text{sim}[i,j])*x[i,j,C[i]] \quad \forall i,j)$$

solve → dapat solusi X2

3. Minimasi max workload:

min max_workload

solve → dapat solusi X3

Optimasi multi-objective:

min $0.05 * (\sum(1 - \sum y[j,k])) + 0.85 * (\sum(1 - sim[i,j]) * x[i,j,k]) +$

0.1 * max_workload

solve → dapat solusi X_moo

Evaluasi:

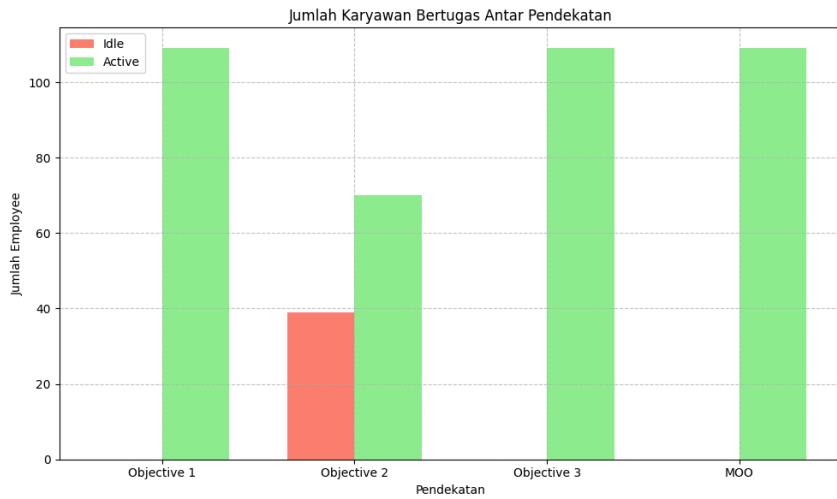
Bandingkan X1, X2, X3, X_moo pada metrik:

- Idle employees
- Rata-rata similarity score
- Variansi workload

End Algoritma

4.4.1 Hasil Objektif Pertama

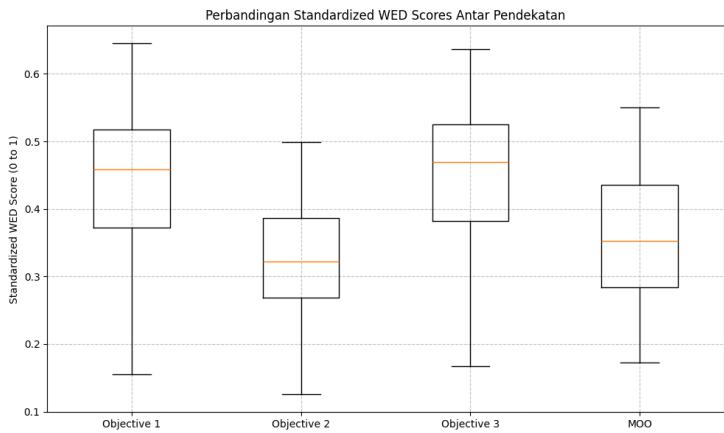
Hasil untuk Objektif pertama, yaitu meminimalkan jumlah karyawan menganggur, menunjukkan performa yang sangat baik dari pendekatan MOO. Dilihat pada gambar 22, MOO berhasil mencapai nol karyawan idle dengan 109 karyawan aktif, setara dengan pendekatan Objektif 1 dan Objektif 3 yang juga mencatatkan nol karyawan idle dan 109 karyawan aktif. Sebaliknya, pendekatan Objektif 2, yang memprioritaskan skor WED untuk dimaksimalkan, menghasilkan 39 karyawan idle dan hanya 70 karyawan aktif, menunjukkan kelemahan signifikan dalam hal pemanfaatan karyawan. Hal ini mengindikasikan bahwa fokus tunggal (*single objective*) pada skor WED cenderung mengorbankan partisipasi karyawan, sehingga banyak karyawan yang tidak mendapatkan tugas. Dengan demikian, MOO terbukti mampu memenuhi Objektif 1 secara optimal, sekaligus menjaga keseimbangan dengan objektif lainnya, menjadikannya solusi yang lebih seimbang dibandingkan pendekatan Objektif 2.



Gambar 23. Hasil Jumlah Karyawan Bertugas Pendekatan MOO

4.4.2 Hasil Objektif Kedua

Objektif kedua bertujuan untuk memaksimalkan skor *Weighted Euclidean Distance* (WED) yang mengukur kecocokan antara keahlian karyawan dan kebutuhan tugas dalam skala 0 hingga 1, dengan 0 mencerminkan kecocokan terburuk dan 1 mencerminkan kecocokan sempurna. Hasil statistik skor WED terstandarisasi menunjukkan bahwa pendekatan *Multi-Objective Optimization* (MOO) berhasil mencapai nilai rata-rata sebesar 0,356776, melampaui pendekatan Objektif 2 (0,326309), meskipun sedikit lebih rendah dibandingkan Objektif 1 (0,441294) dan Objektif 3 (0,449392), dengan nilai median MOO sebesar 0,352536 (lebih baik daripada Objektif 2: 0,322675) dan rentang nilai dari 0,173007 hingga 0,549883 tanpa adanya outliers.



Gambar 24. Hasil Skor WED Pendekatan MOO

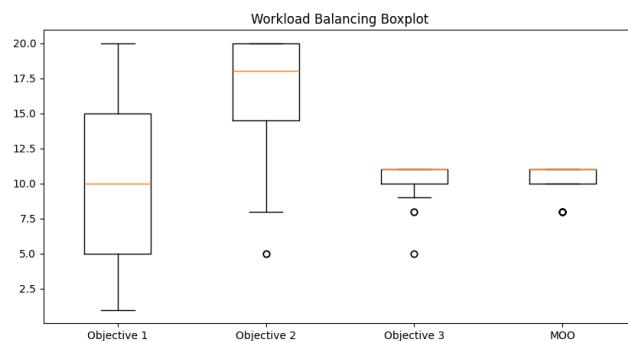
Dalam skala 0-1, nilai rata-rata di bawah 0,5 untuk semua pendekatan menunjukkan bahwa kecocokan keahlian karyawan dengan tugas belum mencapai tingkat optimal, namun hal ini memberikan wawasan berharga mengenai potensi perbaikan dalam penugasan tugas. Penggunaan 65 kategori keterampilan untuk menilai setiap tugas membuka peluang untuk mengevaluasi kecocokan secara menyeluruh, meskipun secara realistik, tidak semua tugas memerlukan evaluasi terhadap seluruh kategori tersebut; pendekatan ini memungkinkan identifikasi keterampilan inti yang benar-benar relevan untuk tugas tertentu, yang dapat menjadi dasar penyederhanaan metrik di masa depan guna meningkatkan efisiensi perhitungan WED.

Keberhasilan MOO dalam melampaui Objektif 2, yang dioptimalkan khusus untuk memaksimalkan WED, menegaskan bahwa pendekatan seimbang yang mempertimbangkan tiga objektif sekaligus mampu memberikan hasil yang lebih baik dibandingkan optimasi tunggal, dengan skor 0,326309 pada Objektif 2 menjadi solusi optimal dalam kerangka optimasi linear menggunakan *Weighted Sum Linear Programming*, memenuhi constraint seperti kapasitas maksimum 20 story points per karyawan dan keunikan penugasan.

Hasil ini menjadi landasan yang kuat untuk eksplorasi lebih lanjut menggunakan pendekatan *Reinforcement Learning* (RL), di mana skor WED yang dicapai dapat dijadikan baseline untuk benchmarking, dengan RL menawarkan potensi untuk meningkatkan kecocokan keahlian secara lebih adaptif melalui desain reward function yang memprioritaskan kategori keterampilan langka seperti "Deep Learning" dan menyeimbangkan trade-off antar objektif secara dinamis, sehingga membuka peluang untuk solusi yang lebih optimal di masa depan.

4.4.3 Hasil Objektif Ketiga

Objektif ketiga, yaitu meminimalisir variansi beban kerja, menunjukkan keunggulan MOO dalam mendistribusikan beban kerja secara merata di antara karyawan.

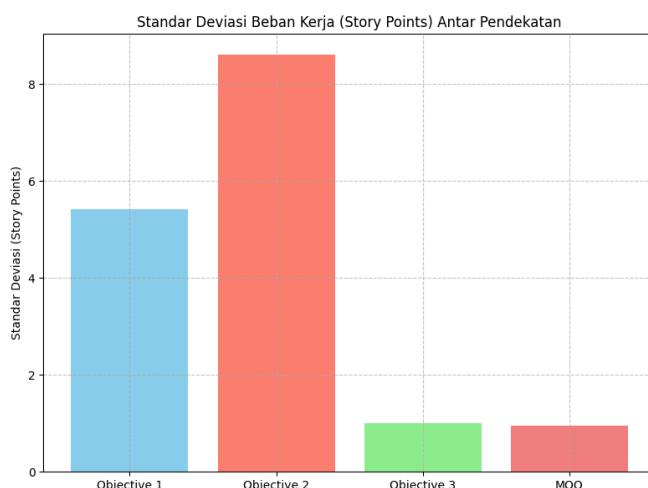


Gambar 25. Boxplot Variansi Pendekatan MOO

Berdasarkan boxplot pada gambar diatas, distribusi beban kerja pada MOO menunjukkan rentang yang sangat sempit dengan median yang stabil, mengindikasikan bahwa sebagian besar karyawan mendapatkan beban kerja yang hampir seragam, tanpa adanya nilai ekstrim yang signifikan. Sebaliknya, boxplot untuk Objektif 1 dan Objektif 2 menunjukkan rentang yang jauh lebih lebar dengan beberapa outlier, menandakan adanya ketidakseimbangan di mana beberapa karyawan mendapatkan beban kerja yang sangat tinggi sementara yang lain sangat

rendah. Objektif 3 juga menunjukkan distribusi yang cukup merata, namun sedikit lebih bervariasi dibandingkan MOO.

Bar chart variansi beban kerja pada gambar 26 mengkonfirmasi temuan ini, dengan MOO mencatatkan nilai variansi terendah, yaitu 0,952893, bahkan melampaui pendekatan Objektif 3 (1,009007) yang secara khusus dioptimasi untuk objektif ini. Sebagai perbandingan, pendekatan Objektif 1 dan Objektif 2 menghasilkan variansi yang jauh lebih tinggi, masing-masing 5,415720 dan 8,602844, mengindikasikan ketidakseimbangan yang signifikan dalam distribusi beban kerja ketika hanya satu objektif diprioritaskan.

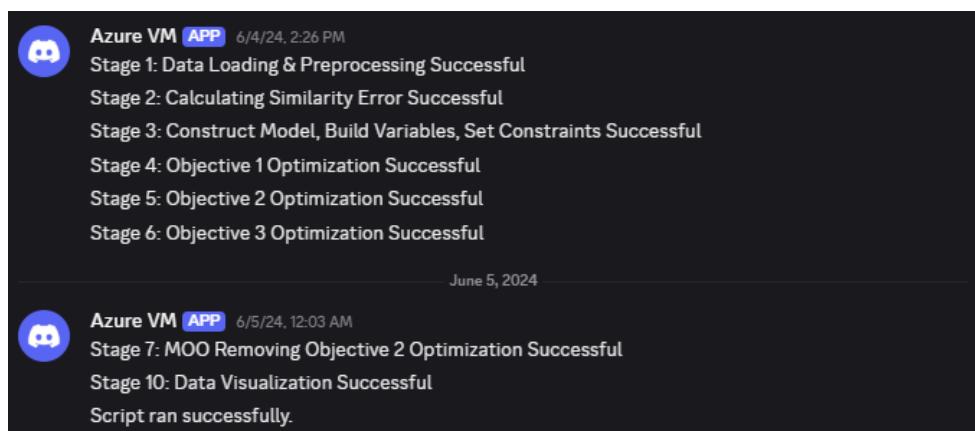


Gambar 26. Bar Chart Variansi Pendekatan MOO

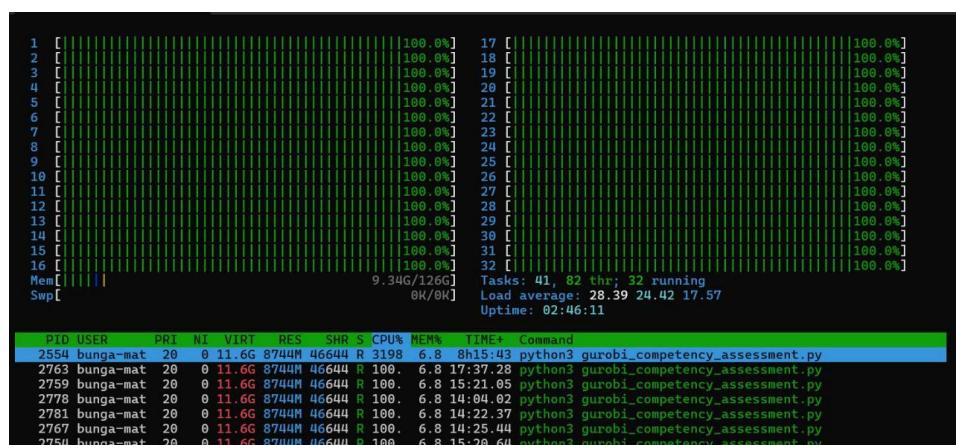
Hasil ini menegaskan bahwa MOO sangat efektif dalam mencapai distribusi beban kerja yang merata, bahkan lebih baik daripada pendekatan yang secara spesifik fokus pada objektif ini. Dengan variansi yang sangat rendah dan distribusi story points yang seragam, MOO memastikan bahwa tidak ada karyawan yang mendapatkan beban kerja berlebihan atau terlalu sedikit, sehingga meningkatkan efisiensi dan keadilan dalam penugasan tugas.

4.4.4 Sumber Komputasi yang Diperlukan

Sesuai gambar 28 dapat dilihat bahwa MOO membutuhkan Virtual Machine dengan spesifikasi 128 GB RAM dan CPU 32 Core. Proses optimasi dilakukan lewat discord. Gambar 27 menunjukkan bahwa proses optimasi membutuhkan waktu 11 jam 30 menit untuk diproses. Sumber Daya ini cukup tinggi, akan dilakukan perbandingan kebutuhan sumberdaya dengan pendekatan RL.



Gambar 27. Notifikasi Otomatis Proses Optimasi MOO



Gambar 28. Virtual Machine untuk Program MOO

4.5 Hasil Pendekatan *Reinforcement Learning*

4.5.1 Implementasi DQN

Implementasi DQN untuk penugasan tugas dilakukan dengan pendekatan RL berbasis pengalaman (*experience replay*). Algoritma ini menggunakan arsitektur *neural network* dengan tiga lapisan tersembunyi (256-128 node) yang memproses representasi state berupa matriks similarity tugas-karyawan, vektor beban kerja, dan fitur global seperti proporsi tugas tersisa. Mekanisme aksi mencakup dua tipe: *assign* (penugasan baru) dan *reassign* (pemindahan tugas), dengan validasi kapasitas karyawan secara real-time.

Sistem reward dirancang *multi-objective* untuk ketiga objektif. Selama pelatihan, eksplorasi diatur melalui ϵ -greedy dengan decay eksponensial, sementara pembelajaran dilakukan secara off-policy melalui sampling batch dari replay memory berkapasitas 50.000 transisi. *Target network* diperbarui secara berkala untuk stabilitas pembelajaran, dengan evaluasi berbasis metrik rata-rata similarity, standar deviasi beban kerja, dan jumlah karyawan idle.

Tabel 22. Pseudocode Implementasi RL

ALGORITMA Pengembangan DQN
Inisialisasi lingkungan TaskAssignmentEnv Muat data karyawan (109) dan tugas (300), normalisasi keterampilan Pad data ke max_tasks=500, max_employees=200, konversi ke tensor Hitung state_matrix dengan top-5 WED-based similarity scores Inisialisasi model DQN dengan FC layers (input: state, output: Q-values) Inisialisasi target_model = model, replay_memory (kapasitas 50000) Inisialisasi optimizer = Adam(lr=3e-4), epsilon = 1.0, gamma = 0.99 FOR episode = 1 hingga 50, DO Reset lingkungan, set observasi = {state_matrix, workloads, features}

```

FOR step = 1 hingga 500, DO
    IF random() < epsilon THEN
        Pilih aksi random (assign/reassign, validasi tugas/workload)
    ELSE
        Hitung Q-values = model(observasi), pilih aksi = argmax(Q)
    END IF
    Eksekusi aksi, dapatkan reward, next_observasi, done
    Simpan transisi di replay_memory
    Sampel batch
    Hitung target = reward + gamma * max(target_model(next_observasi))
    Optimalkan MSE loss
    IF done THEN
        Hentikan episode
    END IF
END FOR
Perbarui epsilon, sinkronkan target_model setiap 1000 langkah
Catat metrik: reward, WED, std workloads, idle employees
END FOR
Simpan model, visualisasikan metrik

END ALGORITMA

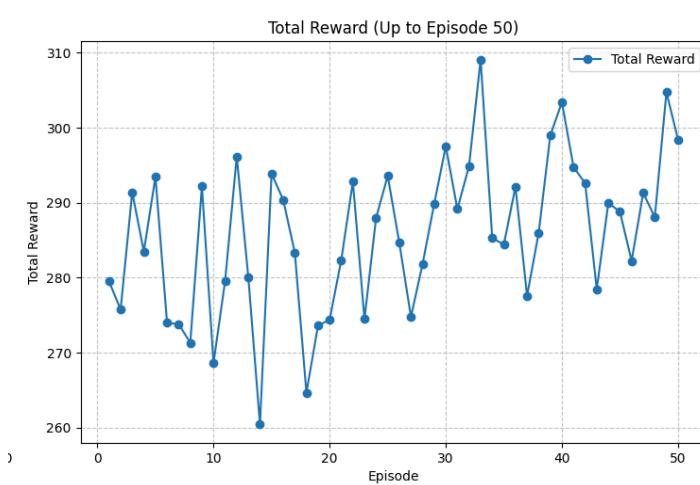
```

4.5.2 Hasil DQN

Proses training DQN dilakukan selama 50 episode akibat keterbatasan komputasi, meskipun idealnya memerlukan lebih banyak episode untuk mencapai konvergensi optimal. Visualisasi hasil training menunjukkan dinamika pembelajaran yang cukup menjanjikan, meskipun masih terdapat fluktuasi yang signifikan.

Gambar 29 memperlihatkan nilai reward gabungan yang mengalami fluktuasi selama 50 episode. Total reward dimulai pada episode 1 dengan nilai 279.5030 dan diakhiri pada episode 50 dengan nilai

298.4210, dengan rentang fluktuasi antara 260.4230 hingga 309.0946. Meskipun terdapat peningkatan pada episode tertentu, seperti episode 33 dengan nilai tertinggi 309.0946, fluktuasi ini menunjukkan bahwa model masih berusaha menyeimbangkan berbagai objektif tanpa konvergensi yang stabil. Peningkatan total reward secara keseluruhan mengindikasikan adanya pembelajaran, tetapi variabilitas yang tinggi menunjukkan bahwa model masih perlu waktu lebih lama untuk mencapai stabilitas optimal.



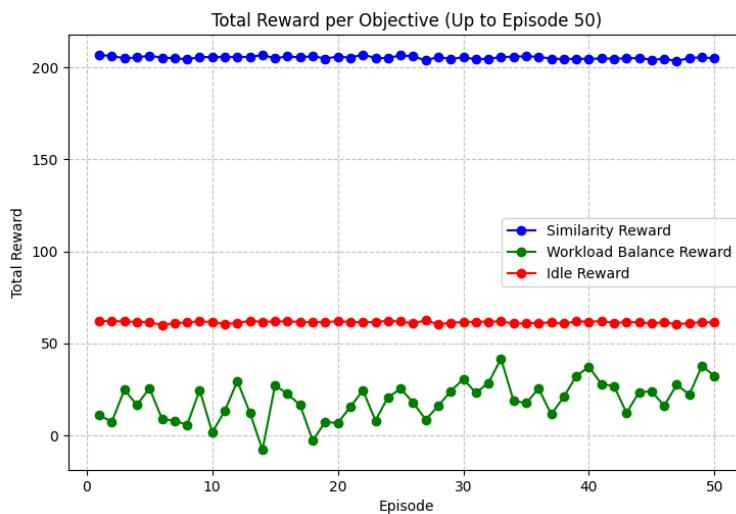
Gambar 29. Nilai Reward Gabungan Pelatihan DQN

Gambar 30 menunjukkan nilai reward masing-masing objektif selama pelatihan. Reward karyawan tidak menganggur (idle reward) relatif stabil pada rentang 60 hingga 62, dengan penurunan kecil pada episode tertentu seperti episode 6 (59.8651) dan episode 47 (60.3651), mencerminkan bahwa model cukup konsisten dalam meminimalkan karyawan menganggur, dengan jumlah karyawan menganggur maksimal hanya 2 pada episode 47.

Sebaliknya, nilai *similarity reward* berfluktuasi ringan di sekitar 204 hingga 206, dimulai dari 206.6109 pada episode 1 dan berakhir di 204.8930 pada episode 50, menunjukkan stabilitas yang cukup baik dalam memaksimalkan kesesuaian antara tugas dan karyawan. Sementara itu,

workload balance reward menunjukkan variabilitas yang lebih besar, dimulai dari 10.8921 pada episode 1, turun ke titik terendah -7.9557 pada episode 14, dan berakhir di 32.2280 pada episode 50, dengan rentang fluktuasi antara -7.9557 hingga 41.6119.

Peningkatan workload balance reward di episode-episode akhir menunjukkan bahwa model mulai belajar untuk menyeimbangkan beban kerja, meskipun trade-off dengan objektif lain masih terlihat jelas. Trade-off ini mencerminkan sifat multi-objective optimization (MOO) dalam RL, di mana model menyesuaikan prioritas antar objektif untuk mencapai keseimbangan. Dibandingkan dengan pendekatan MOO, nilai similarity RL cukup tinggi dan stabil (sekitar 0.65 berdasarkan distribusi WED), tetapi workload balance masih memerlukan optimalisasi lebih lanjut untuk mencapai distribusi yang lebih merata. Pembahasan lebih detail akan diuraikan pada subbab 4.5.



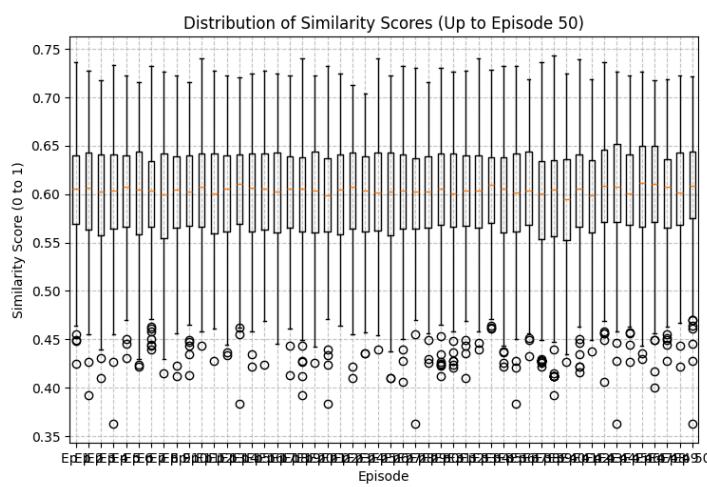
Gambar 30. Nilai Reward Masing Masing Objektif

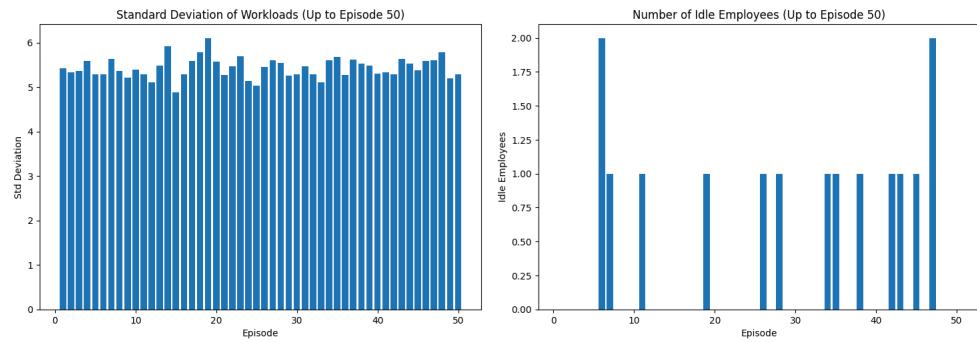
Visualisasi metrik objektif pada Gambar 31 memperkuat temuan ini. Boxplot distribusi WED menunjukkan peningkatan rata-rata dari sedikit di bawah 0.65 menjadi sekitar 0.70 pada episode-episode akhir,

meskipun terdapat beberapa outlier yang lebih rendah di beberapa episode, yang kemungkinan berkontribusi pada fluktuasi reward similarity.

Bar chart variansi beban kerja (standard deviation of workloads) menunjukkan nilai yang relatif stabil, dimulai dari 5.4254 pada episode 1 dan berakhir di 5.2836 pada episode 50, dengan rentang fluktuasi antara 4.8772 hingga 6.1078. Stabilitas ini mengindikasikan bahwa model belum sepenuhnya berhasil mereduksi variansi beban kerja secara signifikan, yang sejalan dengan fluktuasi workload balance reward.

Sementara itu, bar chart jumlah karyawan menganggur menunjukkan performa yang sangat baik, dengan nilai maksimal hanya 2 pada episode 47 dan sebagian besar episode bernilai 0, menandakan efisiensi yang tinggi dalam alokasi sumber daya manusia. Secara keseluruhan, hasil training ini menunjukkan bahwa model RL bergerak menuju optimalitas dengan trade-off yang jelas antara similarity dan workload balance, memberikan fondasi yang kuat untuk perbaikan lebih lanjut jika training diperpanjang.

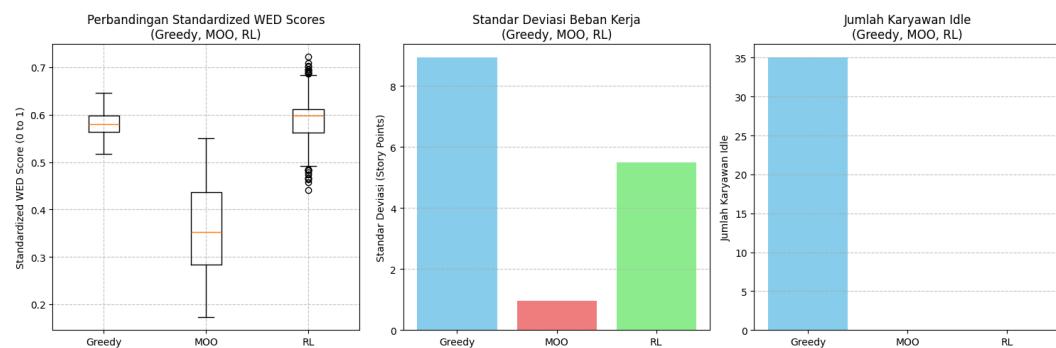




Gambar 31. Metrik Pendekatan RL Selama Pelatihan

4.6 Evaluasi Akhir

Pendekatan Greedy, sebagai acuan dasar, dijalankan pada CPU Intel Xeon 1 core, yang cukup untuk algoritma sederhana ini karena tidak memerlukan komputasi intensif. Pendekatan MOO, sebagai acuan utama, menggunakan solver Gurobi yang dijalankan pada mesin virtual Azure dengan spesifikasi 128 GB RAM dan 32 core CPU, karena Gurobi tidak mendukung optimasi GPU, sehingga memerlukan kapasitas CPU dan memori yang besar untuk menangani kompleksitas optimasi multi-objective. Sementara itu, RL menggunakan DQN yang dilatih selama 50 episode pada T4 GPU melalui Google Colab, memanfaatkan kemampuan GPU untuk mempercepat pelatihan model neural network meskipun terbatas oleh durasi training.

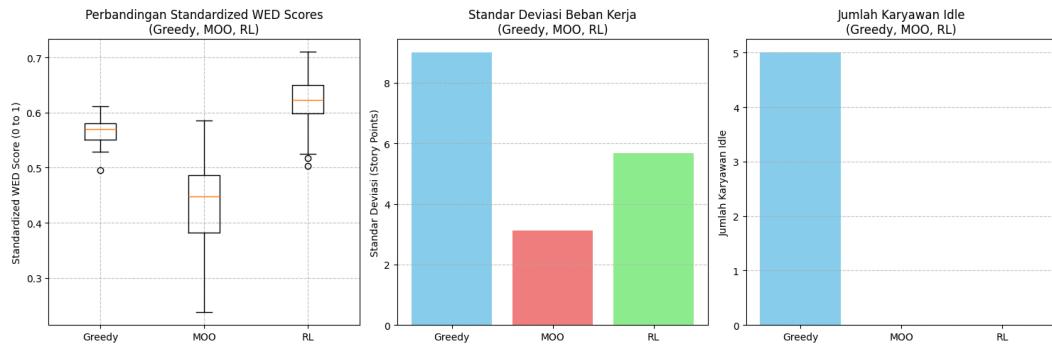


Gambar 32. Hasil Akhir Pendekatan Greedy, MOO, dan RL

Hasil diatas didapatkan pada saat inferensi. Berdasarkan metrik evaluasi, nilai WED yang terstandarisasi menunjukkan bahwa RL memiliki performa terbaik dengan mean 0.589, median 0.598, dan standar deviasi 0.044, diikuti oleh Greedy dengan mean 0.582, median 0.580, dan standar deviasi 0.026, serta MOO dengan mean 0.357, median 0.353, dan standar deviasi 0.091. Hasil ini mengindikasikan bahwa RL mampu mencapai kecocokan keterampilan yang lebih tinggi dibandingkan Greedy dan MOO, dengan variabilitas yang lebih rendah dibandingkan MOO namun sedikit lebih tinggi dibandingkan Greedy, menunjukkan stabilitas yang cukup baik dalam optimasi keterampilan. Performa MOO yang lebih rendah pada metrik WED kemungkinan disebabkan oleh prioritas yang lebih besar pada keseimbangan beban kerja, yang mengorbankan kecocokan keterampilan.

Pada aspek keseimbangan beban kerja, standar deviasi menunjukkan bahwa MOO tetap unggul dengan nilai 0.95 story points, diikuti RL dengan 5.50, dan Greedy dengan 8.93. Hasil ini mencerminkan bahwa MOO memberikan distribusi beban kerja yang paling merata, sementara RL masih perlu dioptimalkan lebih lanjut untuk mendekati performa MOO, dan Greedy memiliki distribusi yang paling tidak seimbang karena tidak mempertimbangkan batasan proyek secara memadai.

Untuk jumlah karyawan menganggur, baik MOO maupun RL berhasil mencapai nol karyawan menganggur, menunjukkan efisiensi maksimal dalam pemanfaatan sumber daya manusia. Sebaliknya, Greedy memiliki 35 karyawan idle, yang menandakan kelemahan signifikan dalam alokasi tugas akibat pendekatan yang terlalu sederhana dan tidak mempertimbangkan semua batasan masalah. Hasil testing berikut dilakukan dengan menggunakan data yang diambil secara random dari data yang digunakan untuk menguji sebanyak 12 karyawan dan 30 tugas. Walaupun RL dilatih dengan lingkungan berbeda, dapat dilihat bahwa RL tetap mampu untuk mengoptimasi dengan baik untuk lingkungan yang berbeda.



Gambar 33. Hasil Testing dengan Lingkungan Baru

Pendekatan RL menunjukkan potensi besar dengan kecocokan keterampilan yang lebih tinggi dan efisiensi alokasi yang setara dengan MOO, meskipun workload balance-nya masih tertinggal. Hasil ini kemungkinan dapat ditingkatkan dengan pelatihan lebih lama, misalnya hingga ratusan atau ribuan episode, yang memerlukan sumber daya komputasi lebih besar. MOO, dengan variasi beban kerja terbaik, tetap menjadi acuan utama dalam hal keseimbangan beban kerja, tetapi membutuhkan sumber daya CPU yang besar dan kurang fleksibel dibandingkan RL dalam adaptasi terhadap perubahan dinamika problem. Greedy, meskipun paling hemat sumber daya, tidak dapat bersaing dalam hal kualitas solusi, terutama karena tidak mematuhi semua batasan masalah, seperti batasan proyek dan distribusi beban kerja.

Implikasi dari perbandingan ini adalah bahwa RL dapat menjadi pendekatan yang lebih adaptif, fleksibel, dan cepat di masa depan dengan optimasi lebih lanjut, terutama dalam meningkatkan workload balance, sementara MOO cocok untuk skenario yang membutuhkan hasil lebih optimal dengan sumber daya komputasi yang memadai.

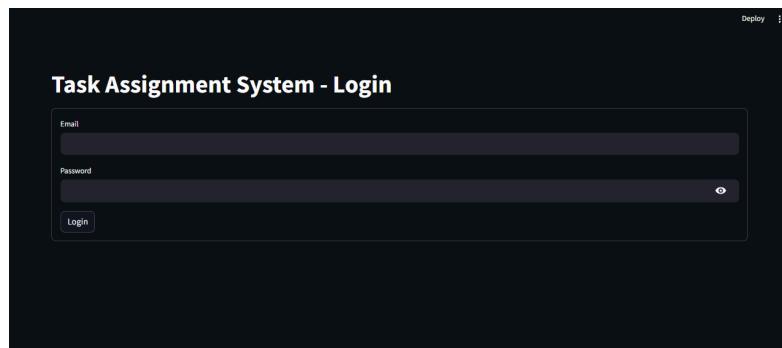
Tabel 23. Hasil Perbandingan Kebutuhan *Hardware*

Pendekatan	Hardware Pelatihan			Waktu Pelatihan	Waktu Inferensi (CPU)
	CPU	GPU	RAM		
Greedy	Intel Xeon	-	12 GB	10 detik	

	1 core				
MOO (MILP)	CPU 32 Core	-	128 GB	11 jam 50 menit	
RL (DQN)	Intel Xeon 1 core	T4 GPU	16 GB	28 menit 32 detik	16 detik

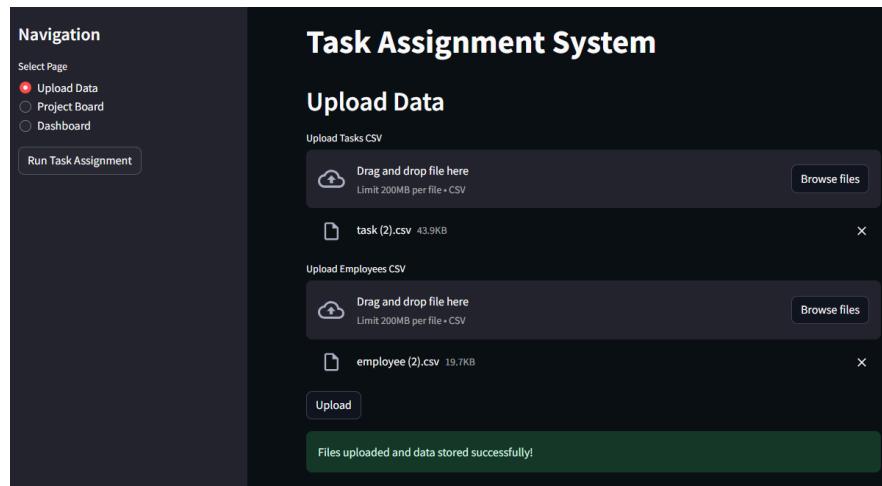
4.7 Hasil Pengembangan Sistem

Halaman login sistem, sebagaimana ditunjukkan pada gambar 34, dirancang dengan tampilan sederhana yang memungkinkan admin untuk mengakses sistem menggunakan email dan kata sandi. Saat ini, fitur pendaftaran belum tersedia untuk pengguna umum, sehingga akses hanya dapat diberikan oleh admin melalui proses registrasi manual.



Gambar 34. Halaman Login

Halaman pertama yang dapat diakses adalah "Upload Data", yang memungkinkan admin mengunggah dataset tugas dan karyawan dalam format CSV. Fitur ini mendukung fleksibilitas input data, dengan jumlah tugas dan karyawan yang dapat disesuaikan sesuai kebutuhan, namun dibatasi hingga maksimum 500 tugas dan 200 karyawan untuk menjaga performa sistem. Batasan ini memastikan bahwa sistem dapat menangani kompleksitas data seperti yang digunakan dalam penelitian ini (300 tugas dan 109 karyawan) tanpa mengalami penurunan kinerja, dengan syarat dataset mematuhi struktur yang telah ditentukan, seperti kategori keterampilan, story points, dan proyek.



Gambar 35. Halaman Unggah Dataset

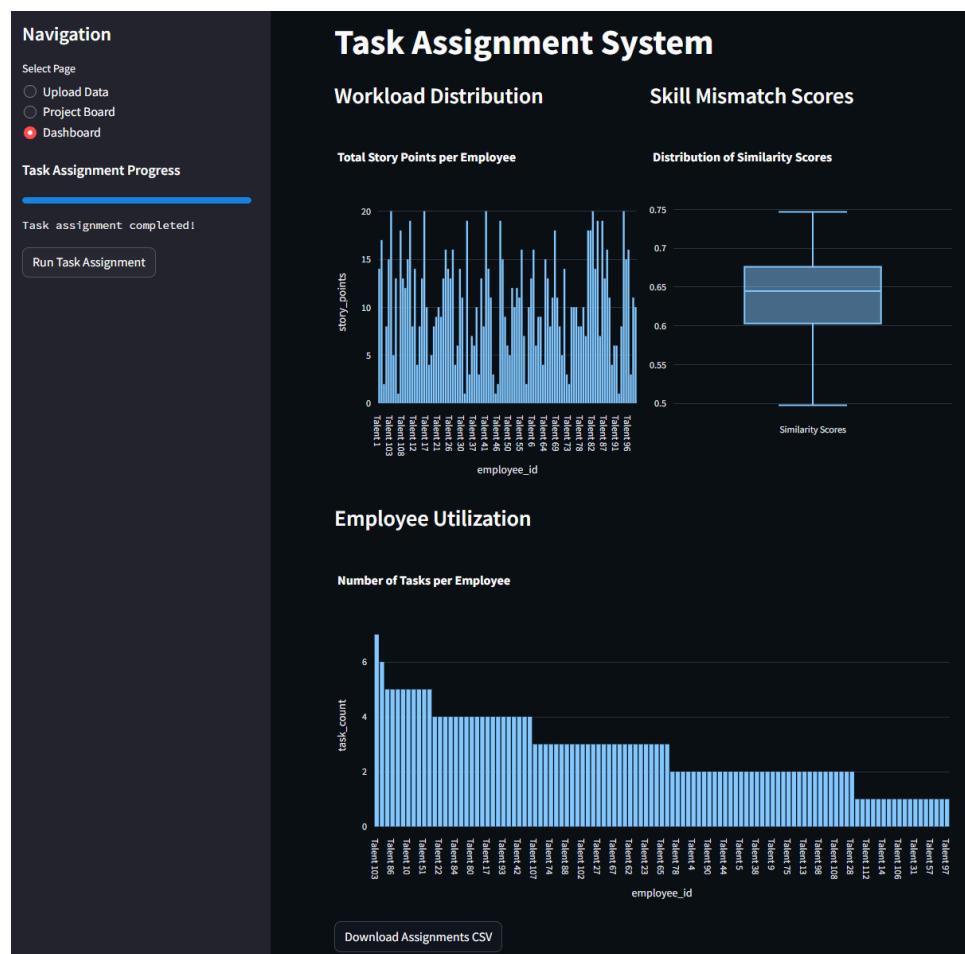
Bagian selanjutnya adalah "Project Backlog", yang menampilkan daftar tugas beserta informasi terkait seperti nama tugas, nama proyek, story points, karyawan yang ditugaskan (PIC), dan status penggerjaan. Fitur ini memungkinkan admin untuk melihat secara langsung bagaimana tugas-tugas dialokasikan kepada karyawan berdasarkan hasil optimasi DQN, memberikan gambaran yang jelas tentang distribusi tugas antar proyek dan karyawan.

Task Name	Project Name	Story Points	PIC	Status
T262	P4	8	Talent 84	To Do
T132	P5	3	Talent 3	To Do
T202	P3	5	Talent 107	To Do
T287	P3	1	Talent 29	To Do
T283	P2	2	Talent 36	To Do
T213	P2	1	Talent 12	To Do
T231	P4	8	Talent 84	To Do
T116	P3	8	Talent 109	To Do
T67	P2	8	Talent 71	To Do

Gambar 36. Halaman Papan Manajemen Proyek

Bagian terakhir adalah halaman *Dashboard*, yang menyediakan visualisasi metrik performa penugasan tugas untuk analisis lebih lanjut.

Visualisasi ini mencakup tiga elemen utama: sebuah boxplot yang menunjukkan distribusi skor kecocokan keterampilan (similarity scores) antara karyawan dan tugas, sebuah bar chart yang menggambarkan bobot beban kerja karyawan (total story points per karyawan), dan bar chart kedua yang menunjukkan pemanfaatan karyawan (jumlah tugas per karyawan). Fitur ini dilengkapi dengan opsi untuk mengunduh hasil penugasan dalam format CSV, memungkinkan admin untuk menyimpan dan menganalisis data lebih lanjut. Secara keseluruhan, sistem ini memberikan solusi yang terintegrasi untuk mendukung penugasan tugas yang optimal, dengan antarmuka yang memudahkan analisis dan pengelolaan data dalam lingkungan kerja dinamis.



Gambar 37. Halaman *Dashboard* Visualisasi Pemanfaatan Karyawan

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan rumusan masalah dan hasil penelitian yang telah dilakukan, dapat ditarik kesimpulan sebagai berikut.

1. Pendekatan RL dengan DQN menunjukkan performa terbaik dalam hal kecocokan keterampilan, dengan skor WED terstandarisasi memiliki rata-rata 0.589, median 0.598, dan standar deviasi 0.044. Pendekatan Greedy menyusul dengan rata-rata 0.582, median 0.580, dan standar deviasi 0.026, sedangkan pendekatan MOO memiliki rata-rata 0.357, median 0.353, dan standar deviasi 0.091. Hasil ini menegaskan bahwa DQN lebih unggul dalam memaksimalkan kecocokan antara keahlian karyawan dan kebutuhan tugas dibandingkan Greedy dan MOO. Meskipun Greedy menunjukkan stabilitas skor WED yang tinggi (standar deviasi rendah), pendekatan ini kurang optimal karena tidak mempertimbangkan batasan proyek secara menyeluruh. Sementara itu, MOO menunjukkan performa yang lebih rendah dalam hal kecocokan keterampilan, dengan variabilitas yang lebih tinggi, kemungkinan karena fokusnya pada keseimbangan beban kerja mengorbankan aspek kecocokan keterampilan.
2. Distribusi keseimbangan beban kerja, yang diukur melalui standar deviasi story points, menunjukkan bahwa MOO memberikan hasil terbaik dengan standar deviasi 0.95, mencerminkan alokasi beban kerja yang sangat merata di antara 109 karyawan. Pendekatan DQN memiliki standar deviasi 5.50, lebih baik dibandingkan Greedy yang mencatatkan 8.93, tetapi masih menunjukkan ketidakseimbangan yang cukup signifikan. Hal ini mengindikasikan bahwa MOO sangat efektif dalam memastikan distribusi beban kerja yang adil, sementara DQN membutuhkan optimasi lebih lanjut untuk mencapai tingkat pemerataan yang serupa, misalnya dengan penyesuaian bobot reward atau pelatihan yang lebih panjang.

3. Jumlah karyawan yang tidak bekerja berhasil diminimalkan oleh pendekatan MOO dan DQN, keduanya mencatatkan nol karyawan menganggur dari total 109 karyawan, menunjukkan efisiensi maksimal dalam pemanfaatan sumber daya manusia. Sebaliknya, pendekatan Greedy meninggalkan 35 karyawan tanpa tugas, menandakan kelemahan signifikan dalam memastikan semua karyawan mendapatkan alokasi tugas, terutama karena tidak mempertimbangkan batasan proyek secara menyeluruh.
4. Perbandingan performa antara DQN dan MOO menunjukkan keunggulan masing-masing pendekatan dalam aspek yang berbeda. DQN lebih unggul dalam kecocokan keterampilan (WED rata-rata 0.589 berbanding 0.357) dan setara dengan MOO dalam meminimalkan karyawan menganggur (0 berbanding 0), tetapi tertinggal dalam keseimbangan beban kerja (standar deviasi 5.50 berbanding 0.95). Dari segi sumber daya komputasi, Greedy dijalankan pada *Google Colab Free* dengan Intel Xeon CPU (1 core fisik) dan RAM 12 GB, menunjukkan kebutuhan komputasi minimal dengan waktu inferensi 10 detik karena sifat algoritmanya yang sederhana. MOO memerlukan mesin virtual Azure dengan Intel Xeon Scalable (32 core fisik) dan RAM 128 GB, dengan waktu pelatihan dan inferensi 11 jam 50 menit menggunakan solver Gurobi yang tidak mendukung GPU. DQN dilatih selama 28 menit 32 detik pada *Google Colab Pro* dengan NVIDIA T4 GPU, Intel Xeon CPU (1 core fisik), dan total RAM 16 GB serta hanya membutuhkan 9 detik untuk inferensi menggunakan CPU Intel Xeon CPU (1 core fisik) dan RAM 12 GB, menunjukkan efisiensi tinggi pada fase penerapan. Perbandingan waktu dipengaruhi oleh perbedaan kapasitas hardware, di mana Azure VM (32 core) jauh lebih kuat dibandingkan Colab Free (1 core) dan Colab Pro (2 core). Secara keseluruhan, DQN menawarkan solusi yang lebih adaptif, fleksibel, dan cepat pada tahap inferensi, dengan potensi peningkatan performa melalui pelatihan lebih lama, sementara MOO tetap unggul untuk distribusi beban kerja yang merata, meskipun membutuhkan sumber daya komputasi yang lebih besar.

5.2 Saran

Berdasarkan hasil penelitian, berikut adalah beberapa saran untuk pengembangan lebih lanjut:

1. Optimasi Pelatihan DQN dengan cara perpanjang pelatihan DQN hingga ribuan episode untuk mencapai konvergensi yang lebih baik, terutama dalam mengoptimalkan keseimbangan beban kerja. Hal ini dapat meningkatkan performa DQN agar lebih seimbang dalam ketiga objektif, mendekati atau melampaui MOO dalam distribusi beban kerja. Yang tentunya dapat dilakukan dengan kemampuan komputasi yang lebih baik.
2. Penyempurnaan Strategi DQN dengan mengembangkan *reward function* yang lebih seimbang pada DQN dan *state* yang lebih sederhana dan representatif, misalnya dengan memberikan prioritas lebih besar pada keseimbangan beban kerja atau memanfaatkan data keterampilan langka untuk meningkatkan kecocokan secara lebih cerdas, sehingga *trade-off* antar objektif dapat lebih terkontrol.
3. Integrasi Pendekatan Hibrid dengan mengintegrasikan keunggulan MOO dan DQN melalui pendekatan hibrid, seperti menggunakan hasil DQN sebagai panduan awal untuk penyelesaian MOO, dapat menghasilkan solusi yang menggabungkan distribusi beban kerja optimal dari MOO dan fleksibilitas adaptif dari DQN.
4. Explorasi berbagai pendekatan RL lainnya seperti PPO untuk permasalahan MILP seperti pada penelitian ini. Pendekatan lain mungkin dapat memberikan pembelajaran yang lebih baik pada RL untuk *action space* yang cukup besar.

Saran-saran ini diharapkan dapat meningkatkan efektivitas penugasan tugas, baik dari segi kualitas solusi maupun efisiensi komputasi, sehingga dapat diterapkan secara lebih luas dalam lingkungan kerja dinamis.

DAFTAR PUSTAKA

- Alablani, I., & Alenazi, M. (2023). DQN-GNN-Based User Association Approach for Wireless Networks. <https://doi.org/10.3390/math11204286>
- Barrett, T. D., & et al. (2020, January 31). Exploratory Combinatorial Optimization with Reinforcement Learning. *Proceedings of Thirty-fourth AAAI conference on artificial intelligence*, 3243-3250. <https://doi.org/10.48550/arXiv.1909.04063>
- Blockeel, H. (2023). Decision trees: from efficient prediction to responsible AI. *Frontiers in Artificial Intelligence*, 8. <https://doi.org/10.3389/frai.2023.1124553>
- Bonz, J. (2021). Application of a multi-objective multi traveling salesperson problem with time windows. *Public Transp.*, 13, 35-57. <https://doi.org/10.1007/s12469-020-00258->
- Broek, J. V. (2008). Timetabling problems at the TU Eindhoven. <https://doi.org/10.1016/j.ejor.2008.04.038>
- Domenech, B., & Lusa, A. (2016). A MILP model for the teacher assignment problem considering teachers' preferences. *Innovative Application of O.R.* <https://doi.org/10.1016/j.ejor.2015.08.057>
- Hamilton, W. L. (2020). *Graph Representation Learning*. Springer International Publishing.
- Hasselt, H. V., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. 31.

- Hessel, M. (2018). Rainbow: Combining improvements in deep reinforcement learning. *32*.
- Hoffman, K. L., & Ralphs, T. K. (2012). Integer and Combinatorial Optimization. https://www.academia.edu/15152674/Integer_and_Combinatorial_Optimization
- Hubbs, C. D., & et al. (2020, October 4). A deep reinforcement learning approach for chemical production scheduling. *Computers & Chemical Engineering*, *141*. <https://doi.org/10.1016/j.compchemeng.2020.106982>
- Mnih, V., Kavukcuoglu, K., & David, S. (2015). Human-level control through deep reinforcement learning. <https://doi.org/10.1038/nature14236>
- Mnih, V., Kavukcuoglu, K., & Silver, D. (2013). Playing Atari with Deep Reinforcement Learning. <https://arxiv.org/pdf/1312.5602>
- O'Malley, C., & et. al. (2023, November 1). Reinforcement learning and mixed-integer programming for power plant scheduling in low carbon systems: Comparison and hybridisation. *Applied Energy*, *349*. <https://doi.org/10.1016/j.apenergy.2023.121659>
- Rao, R. V., & Singh, D. (2012). Weighted Euclidean distance based approach as a multiple attribute decision making method for plant or facility layout design selection. *International Journal of Industrial Engineering Computations*, *3*. [10.5267/j.ijiec.2012.01.003](https://doi.org/10.5267/j.ijiec.2012.01.003)
- Sutton, R. S., & Barto, A. G. (2013). *Reinforcement Learning An Introduction* (Second Edition ed.). The MIT Press.
- <http://incompleteideas.net/book/RLbook2020.pdf>

- Wang, Z., Bapst, V., & Heess, N. (2016). Sample efficient actor-critic with experience replay. <https://doi.org/10.48550/arXiv.1611.01224>
- Yang, X.-S. (2008). *Introduction to Mathematical Optimization: From Linear Programming to Metaheuristics*. Cambridge International Science Publishing.
- mrce.in/ebooks/Mathematical%20Optimization%20Introduction.pdf
- Zhao, Y. (2020). Preference-aware Task Assignment in Spatial Crowdsourcing: from Individuals to Groups.
https://zheng-kai.com/paper/tkde_2020_zhao.pdf
- Zhu, J., Wu, F., & Zhao, J. (2021). An Overview of the Action Space for Deep Reinforcement Learning. *2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*.
<https://doi.org/10.1145/3508546.3508598>

LAMPIRAN

Lampiran 1. Dataset Karyawan & Tugas

task_id	project_id	story_points	Mathematics .Linear Algebra	Mathematics .Differential Equations	Mathematics .Optimization Technique	Mathematics .Calculus	Mathematics .Combinatorics & Graph	Statistics & Probabilities. Statistics	...	MLOPS.Ethical AI & Bias Mitigation
T1	P5	8	2	1	1	0	1	4	...	0
T2	P1	1	3	2	2	3	2	5	...	5
T3	P5	2	3	5	4	5	5	1	...	

No	employee_id	Role	Mathematics .Linear Algebra	Mathematics .Differential Equations	Mathematics .Optimization Technique	Mathematics .Calculus	Mathematics .Combinatorics & Graph	Statistics & Probabilities. Statistics	...	MLOPS.Ethical AI & Bias Mitigation
1	Talent 1	Data Scientist	1	1	2	1	2	3	...	2
2	Talent 2	Data Scientist	4	4	4	4	3	5	...	1
3	Talent 3	Data Analyst	3	2	2	3	2	3	...	2

Dikarenakan ukuran dataset terlalu besar untuk dilampirkan secara langsung, maka dataset akan dilampirkan dalam bentuk tautan yang dapat diakses sebagai berikut:

<https://bit.ly/DatasetKaryawanTugas>

Lampiran 2. Bukti Izin Penggunaan Dataset

SURAT PERNYATAAN IZIN PENGGUNAAN DATASET

Yang bertanda tangan di bawah ini:

Nama : Muhammad Rayes Fitra Gifari
Alamat : Jl Raya Cinunuk Cileunyi, Kab Bandung
Kontak : rayes.gifari@telkom.co.id
Status : Data Scientist

Selanjutnya disebut sebagai **PIHAK PERTAMA**

Nama : I Gusti Putu Wisnu Wardhana
Alamat : Jl. Hayam Wuruk 10/15, Kayumas Kaja, Dangin Puri, Denpasar Timur
Kontak : +62 812-8961-7416
NIM: : 2108561064
Program Studi : Informatika
Fakultas : Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas : Universitas Udayana

Selanjutnya disebut sebagai **PIHAK KEDUA**, dengan ini menyatakan bahwa:

1. PIHAK PERTAMA adalah pemilik dataset dan terlibat dalam kegiatan magang PIHAK KEDUA di PT. Telkom Indonesia STO Kebayoran Baru pada periode 16 Februari hingga 30 Juni 2024.
2. PIHAK PERTAMA memberikan izin kepada PIHAK KEDUA untuk menggunakan dataset yang terlampir pada surat pernyataan ini untuk keperluan penyusunan skripsi berjudul "Optimasi Penugasan Proyek yang Adaptif terhadap Dinamika Tenaga Kerja dengan Reinforcement Learning dan Multi Objective Optimization".
3. Dataset tersebut telah dipastikan oleh PIHAK PERTAMA tidak mengandung informasi rahasia yang dilindungi, sebagaimana dengan bukti terlampir juga dikonfirmasi oleh Internship Management PT. Telkom Indonesia lewat email tanggal 18 Maret 2025 pukul 10:55 WITA.

Surat pernyataan ini kami buat dengan sebenar-benarnya dan dapat digunakan sebagai bukti izin penggunaan dataset sesuai kebutuhan akademik.

Jakarta, 26 Maret 2025

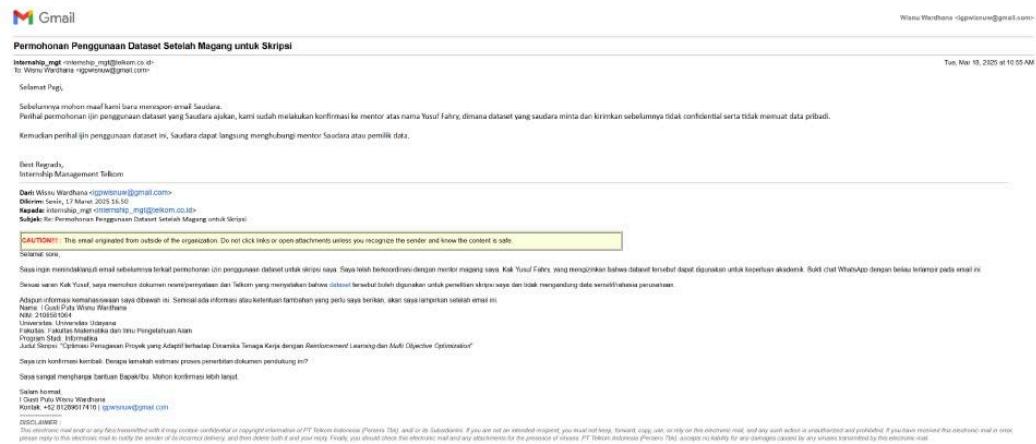
Muhammad Rayes Fitra Gifari
PIHAK PERTAMA

Denpasar, 26 Maret 2025

I Gusti Putu Wisnu Wardhana
PIHAK KEDUA

Lampiran

Bukti email konfirmasi perizinan penggunaan dataset dengan Internship Management Telkom



Tautan akses dataset

<https://docs.google.com/spreadsheets/d/1UEEJFI-YEaWTEEbKOoW9Jg61M0LGwGZujEXC8lj9rTQ/edit?usp=sharing>

Lampiran 3. Kode DQN: Impor data dan tentukan parameter global

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import gym
from gym import spaces
import random
from collections import deque
import time
from torch.cuda.amp import GradScaler as AmpGradScaler
import matplotlib.pyplot as plt
import seaborn as sns
import uuid
import os

# Tentukan perangkat komputasi: GPU jika tersedia, jika tidak
gunakan CPU
device = torch.device("cuda" if torch.cuda.is_available()
else "cpu")
print(f"Menggunakan perangkat: {device}")

# Definisi variabel global untuk konfigurasi pelatihan dan
lingkungan
# MAX_TASKS = 500 # Jumlah maksimum tugas
```

```

# MAX_EMPLOYEES = 200 # Jumlah maksimum karyawan
NUM_EPISODES = 50 # Jumlah episode pelatihan
LEARNING_RATE = 3e-4 # Laju pembelajaran untuk optimisasi
jaringan saraf
INITIAL_EPSILON = 1.0 # Nilai awal epsilon untuk eksplorasi
epsilon-greedy
MIN_EPSILON = 0.1 # Nilai minimum epsilon
GAMMA = 0.99 # Faktor diskon untuk perhitungan Q-value
MEMORY_SIZE = 50000 # Kapasitas memori replay
BATCH_SIZE = 64 # Ukuran batch untuk pembelajaran
TARGET_UPDATE_FREQ = 1000 # Frekuensi pembaruan jaringan
target
DQN_HIDDEN_DIM = 256 # Dimensi lapisan tersembunyi jaringan
DQN
MAX_WORKLOAD = 20 # Batas maksimum beban kerja per karyawan
ALPHA_WED = 0.5 # Parameter untuk Weighted Euclidean
Distance (WED)
REASSIGNMENT_BONUS = 5.0 # Bonus untuk reassignment yang
meningkatkan keseimbangan
REASSIGNMENT_THRESHOLD = 5.0 # Ambang deviasi standar untuk
memicu reassignment
MAX_REASSIGN_PER_TASK = 3 # Batas maksimum reassignment per
tugas
STEP_PENALTY = -0.01 # Penalti per langkah untuk mendorong
efisiensi
COMPLETION_BONUS = 50.0 # Bonus saat semua tugas selesai
dialokasikan

SAVE_EPISODES = [20] # Episode untuk menyimpan checkpoint
MAX_STEPS_PER_EPISODE = MAX_TASKS # Maksimum langkah per
episode
TOTAL_STEPS = NUM_EPISODES * MAX_STEPS_PER_EPISODE # Total
langkah pelatihan
DECAY_RATE_EPS = -np.log(MIN_EPSILON / INITIAL_EPSILON) /
TOTAL_STEPS # Laju peluruhan epsilon

# Path untuk data dan penyimpanan model
# task_data_path =
#/content/drive/MyDrive/Skripsi/Resources/Datasets/fixed_
task.csv'
# employee_data_path =
#/content/drive/MyDrive/Skripsi/Resources/Datasets/fixed_
employee.csv'
model_save_path =
'/content/drive/MyDrive/Skripsi/reassignment_finalcode_dqn_mo
del.pth'
agent_save_path =
'/content/drive/MyDrive/Skripsi/reassignment_finalcode_dqn_ag
ent_params.pth'

# Direktori untuk menyimpan log
log_dir = '/content/drive/MyDrive/Skripsi/Logs/'
os.makedirs(log_dir, exist_ok=True)

```

Lampiran 4. Kode DQN: Lingkungan sebagai kelas TaskAssignmentEnv

```
# Kelas lingkungan untuk simulasi penugasan tugas
class TaskAssignmentEnv(gym.Env):
    def __init__(self, task_data, employee_data,
max_tasks=MAX_TASKS, max_employees=MAX_EMPLOYEES):
        super(TaskAssignmentEnv, self).__init__()
        # Inisialisasi jumlah tugas dan karyawan
        self.num_tasks = len(task_data)
        self.num_employees = len(employee_data)
        self.max_tasks = max_tasks
        self.max_employees = max_employees
        self.max_workload = MAX_WORKLOAD

        # Memuat dan memproses data tugas dan karyawan
        task_subset = task_data.iloc[:self.num_tasks]
        employee_subset =
employee_data.iloc[:self.num_employees]
        skill_columns = [col for col in task_data.columns if
col not in ['task_id', 'story_points', 'project_id']]
        # Mengonversi ID proyek menjadi indeks unik
        project_id_map = {pid: idx + 1 for idx, pid in
enumerate(task_subset['project_id'].unique())}
        project_ids =
task_subset['project_id'].map(project_id_map).values
        # Normalisasi data keterampilan dan story points
        task_skills_df =
task_subset[skill_columns].fillna(0).astype(np.float32)
        employee_skills_df =
employee_subset[skill_columns].fillna(0).astype(np.float32)
        story_points_df =
task_subset['story_points'].astype(np.float32)

        # Konversi data ke tensor dan lakukan padding untuk
        ukuran maksimum
        self.task_skills = F.pad(
            torch.tensor(task_skills_df.values / 5.0,
dtype=torch.float32, device=device),
            (0, 0, 0, max_tasks - self.num_tasks),
            'constant', 0
        )
        self.employee_skills = F.pad(
            torch.tensor(employee_skills_df.values / 5.0,
dtype=torch.float32, device=device),
            (0, 0, 0, max_employees - self.num_employees),
            'constant', 0
        )
        self.story_points = F.pad(
            torch.tensor(story_points_df.values,
dtype=torch.float32, device=device),
            (0, max_tasks - self.num_tasks),
```

```

        'constant', 0
    )
    self.project_ids = F.pad(
        torch.tensor(project_ids, dtype=torch.long,
device=device),
        (0, max_tasks - self.num_tasks),
        'constant', -1
    )
    self.total_story_points =
torch.sum(self.story_points[:self.num_tasks]).item()
    self.num_skills = len(skill_columns)

    # Definisi ruang observasi dan aksi
    self.observation_space = spaces.Dict({
        'state_matrix': spaces.Box(low=0, high=1,
shape=(max_tasks, 5), dtype=np.float32),
        'workload_matrix': spaces.Box(low=0,
high=MAX_WORKLOAD, shape=(max_employees,), dtype=np.float32),
        'global_features': spaces.Box(low=0, high=1,
shape=(3,), dtype=np.float32),
        'task_mask': spaces.Box(low=0, high=1,
shape=(max_tasks,), dtype=np.float32)
    })
    self.action_space = spaces.MultiDiscrete([2,
self.num_tasks, self.num_employees])
    self.reset()

    # Mengatur ulang lingkungan untuk episode baru
    def reset(self):
        # Inisialisasi beban kerja, penugasan, dan status
        projek
        self.workloads = torch.zeros(self.max_employees,
device=device, dtype=torch.float32)
        self.assignments = torch.full((self.max_tasks,), -1,
dtype=torch.long, device=device)
        self.remaining_tasks = torch.arange(self.num_tasks,
device=device, dtype=torch.long)
        self.employee_projects =
torch.full((self.max_employees,), -1, dtype=torch.long,
device=device)
        self.wed_scores = [] # Menyimpan skor WED untuk
analisis
        self.reassign_counts = torch.zeros(self.max_tasks,
device=device, dtype=torch.long) # Melacak reassignment per
tugas
        self.total_reassigs = 0 # Total reassignment dalam
episode
        self._initialize_state_matrix()
        return self._get_obs()

    # Inisialisasi matriks status berdasarkan kesamaan
keterampilan
    def _initialize_state_matrix(self):

```

```

        # Menghitung kesamaan antara keterampilan tugas dan
        # karyawan menggunakan WED
        task_skills_expanded =
        self.task_skills[:self.num_tasks].unsqueeze(1).expand(-1,
        self.max_employees, -1)
        emp_skills_expanded =
        self.employee_skills.unsqueeze(0).expand(self.num_tasks, -1,
        -1)
        diff = emp_skills_expanded - task_skills_expanded
        weights = 1 / (1 + ALPHA_WED * torch.maximum(diff,
        torch.tensor(0.0, device=device, dtype=torch.float32)))
        mask =
        self.task_skills[:self.num_tasks].unsqueeze(1).expand(-1,
        self.max_employees, -1) > 0
        weighted_diff = weights * mask * (emp_skills_expanded
        - task_skills_expanded) ** 2
        wed = torch.sqrt(torch.sum(weighted_diff, dim=2))
        num_skills_per_task =
        torch.sum(self.task_skills[:self.num_tasks] > 0,
        dim=1).unsqueeze(1).expand(-1, self.max_employees)
        wed_worst = torch.sqrt(num_skills_per_task * (1.0 ** 2))
        wed_worst = torch.where(num_skills_per_task > 0,
        wed_worst, torch.tensor(1.0, device=device,
        dtype=torch.float32))
        similarity_matrix = (1 - (wed /
        wed_worst)).to(dtype=torch.float32)

        # Ambil 5 nilai kesamaan tertinggi untuk setiap tugas
        top_k = 5
        self.state_matrix = torch.topk(similarity_matrix,
        k=top_k, dim=1).values
        self.state_matrix = F.pad(self.state_matrix, (0, 0,
        0, self.max_tasks - self.num_tasks), 'constant', 0)

        # Memperbarui matriks status untuk tugas tertentu setelah
        # penugasan
        def _update_state_matrix(self, task_idx, emp_idx):
            task_skills_expanded =
            self.task_skills[task_idx].unsqueeze(0).expand(self.max_employees,
            -1)
            emp_skills_expanded =
            self.employee_skills.expand(self.max_employees, -1)
            diff = emp_skills_expanded - task_skills_expanded
            weights = 1 / (1 + ALPHA_WED * torch.maximum(diff,
            torch.tensor(0.0, device=device, dtype=torch.float32)))
            mask = task_skills_expanded > 0
            weighted_diff = weights * mask * (emp_skills_expanded
            - task_skills_expanded) ** 2
            wed = torch.sqrt(torch.sum(weighted_diff, dim=1))
            num_skills = torch.sum(mask.float(), dim=1)
            wed_worst = torch.sqrt(num_skills * (1.0 ** 2)) if
            num_skills[0] > 0 else 1.0

```

```

        similarities = 1 - (wed / wed_worst)
        top_k_similarities = torch.topk(similarities, k=5,
dim=0).values
        self.state_matrix[task_idx] = top_k_similarities

    # Mengembalikan observasi saat ini
    def _get_obs(self):
        workload_matrix = self.workloads
        prop_remaining = len(self.remaining_tasks) /
self.num_tasks if self.num_tasks > 0 else 0.0
        num_idle =
torch.sum(self.workloads[:self.num_employees] == 0).item()
        prop_idle = num_idle / self.num_employees if
self.num_employees > 0 else 0.0
        std_workload =
torch.std(self.workloads[:self.num_employees]).item() /
self.max_workload
        global_features = torch.tensor([prop_remaining,
prop_idle, std_workload], device=device, dtype=torch.float32)
        task_mask = torch.zeros(self.max_tasks,
device=device, dtype=torch.float32)
        if len(self.remaining_tasks) > 0:
            task_mask[self.remaining_tasks] = 1
        return {
            'state_matrix': self.state_matrix,
            'workload_matrix': workload_matrix,
            'global_features': global_features,
            'task_mask': task_mask
        }

    # Melakukan satu langkah dalam lingkungan
    def step(self, action):
        action_type, task_idx, emp_idx = action
        done = False
        reward = {'similarity': 0.0, 'workload': 0.0, 'idle': 0.0}
        info = {'action_valid': True}

        # Validasi indeks tugas dan karyawan
        if emp_idx < 0 or emp_idx >= self.num_employees or
task_idx < 0 or task_idx >= self.num_tasks:
            reward['similarity'] = -1.0
            info['action_valid'] = False
            info['invalid_action'] = 'Indeks tugas atau
karyawan tidak valid'
            return self._get_obs(), reward, done, info

        # Hitung deviasi standar beban kerja sebelum aksi
        current_std =
torch.std(self.workloads[:self.num_employees]).item()

        # Penugasan baru (action_type = 0)
        if action_type == 0:

```

```

        if task_idx not in self.remaining_tasks:
            reward['similarity'] = -1.0
            info['action_valid'] = False
            info['invalid_action'] = 'Tugas sudah
ditugaskan'
            return self._get_obs(), reward, done, info
    task_project = self.project_ids[task_idx]
    emp_project = self.employee_projects[emp_idx]
    current_workload = self.workloads[emp_idx]
    if emp_project != -1 and emp_project !=
task_project:
        reward['similarity'] = -1.0
        info['action_valid'] = False
        info['invalid_action'] = 'Proyek berbeda'
        return self._get_obs(), reward, done, info
        if current_workload + self.story_points[task_idx]
> self.max_workload:
            reward['similarity'] = -1.0
            info['action_valid'] = False
            info['invalid_action'] = 'Melebihi beban
kerja'
            return self._get_obs(), reward, done, info
    was_idle = (self.workloads[emp_idx] == 0)
    self.assignments[task_idx] = emp_idx
    self.workloads[emp_idx] +=
self.story_points[task_idx]
    self.remaining_tasks =
self.remaining_tasks[self.remaining_tasks != task_idx]
    if self.employee_projects[emp_idx] == -1:
        self.employee_projects[emp_idx] =
task_project
        self._update_state_matrix(task_idx, emp_idx)

        # Reassignment (action_type = 1)
    elif action_type == 1:
        if task_idx in self.remaining_tasks:
            reward['similarity'] = -1.0
            info['action_valid'] = False
            info['invalid_action'] = 'Tugas belum
ditugaskan'
            return self._get_obs(), reward, done, info
        if self.reassign_counts[task_idx] >=
MAX_REASSIGN_PER_TASK:
            reward['similarity'] = -1.0
            info['action_valid'] = False
            info['invalid_action'] = 'Batas reassignment
tercapai'
            return self._get_obs(), reward, done, info
    current_emp_idx = self.assignments[task_idx]
    if current_emp_idx == emp_idx:
        reward['similarity'] = -1.0
        info['action_valid'] = False
        info['invalid_action'] = 'Tugas sudah

```

```

ditugaskan ke karyawan ini'
        return self._get_obs(), reward, done, info
    task_project = self.project_ids[task_idx]
    emp_project = self.employee_projects[emp_idx]
    current_workload = self.workloads[emp_idx]
    if emp_project != -1 and emp_project != task_project:
        reward['similarity'] = -1.0
        info['action_valid'] = False
        info['invalid_action'] = 'Proyek berbeda
untuk karyawan baru'
        return self._get_obs(), reward, done, info
    if current_workload + self.story_points[task_idx] > self.max_workload:
        reward['similarity'] = -1.0
        info['action_valid'] = False
        info['invalid_action'] = 'Melebihi beban
kerja karyawan baru'
        return self._get_obs(), reward, done, info
    self.workloads[current_emp_idx] -= self.story_points[task_idx]
    emp_tasks = (self.assignments == current_emp_idx) & (self.project_ids == task_project)
    emp_tasks[task_idx] = False
    if not torch.any(emp_tasks):
        self.employee_projects[current_emp_idx] = -1
        was_idle = (self.workloads[emp_idx] == 0)
        self.assignments[task_idx] = emp_idx
        self.workloads[emp_idx] += self.story_points[task_idx]
        if self.employee_projects[emp_idx] == -1:
            self.employee_projects[emp_idx] =
task_project
            self.reassign_counts[task_idx] += 1
            self.total_reassings += 1
            self._update_state_matrix(task_idx, emp_idx)

    else:
        reward['similarity'] = -1.0
        info['action_valid'] = False
        info['invalid_action'] = 'Tipe aksi tidak valid'
        return self._get_obs(), reward, done, info

    # Hitung deviasi standar setelah aksi dan berikan
reward keseimbangan
    new_std =
torch.std(self.workloads[:self.num_employees]).item()
    std_change = new_std - current_std
    if std_change < 0:
        reward['workload'] += 0.5
    elif std_change > 0:
        reward['workload'] -= 0.3

```

```

# Hitung reward kesamaan berdasarkan WED
task_skills_t = self.task_skills[task_idx]
emp_skills_e = self.employee_skills[emp_idx]
diff = emp_skills_e - task_skills_t
weights = 1 / (1 + ALPHA_WED * torch.maximum(diff,
torch.tensor(0.0, device=device, dtype=torch.float32)))
mask = task_skills_t > 0
weighted_diff = weights[mask] * (emp_skills_e[mask] -
task_skills_t[mask]) ** 2
wed = torch.sqrt(torch.sum(weighted_diff))
num_skills = torch.sum(mask.float())
wed_worst = torch.sqrt(num_skills * (1.0 ** 2)) if
num_skills > 0 else 1.0
similarity = 1 - (wed / wed_worst) if wed_worst > 0
else 0.0
info['similarity'] = similarity.item()
self.wed_scores.append(wed.item())

if action_type == 0:
    reward['similarity'] = similarity.item()
    if was_idle:
        reward['idle'] = 0.1
    else:
        reassign_decay = 0.5 **
self.reassign_counts[task_idx].item()
        if std_change < 0:
            reward['similarity'] = (REASSIGNMENT_BONUS *
abs(std_change)) * reassign_decay
        else:
            reward['similarity'] = 0.0

reward['similarity'] += STEP_PENALTY

# Cek apakah semua tugas telah ditugaskan
if len(self.remaining_tasks) == 0:
    done = True
    reward['similarity'] += COMPLETION_BONUS
    assigned_tasks = torch.where(self.assignments !=
-1)[0]
    total_similarity = 0.0
    for task_idx in assigned_tasks:
        emp_idx = self.assignments[task_idx].item()
        task_skills_t = self.task_skills[task_idx]
        emp_skills_e = self.employee_skills[emp_idx]
        diff = emp_skills_e - task_skills_t
        weights = 1 / (1 + ALPHA_WED *
torch.maximum(diff, torch.tensor(0.0, device=device,
dtype=torch.float32)))
        mask = task_skills_t > 0
        weighted_diff = weights[mask] *
(emp_skills_e[mask] - task_skills_t[mask]) ** 2
        wed = torch.sqrt(torch.sum(weighted_diff))
        num_skills = torch.sum(mask.float())

```

```

        wed_worst = torch.sqrt(num_skills * (1.0 ** 2)) if num_skills > 0 else 1.0
                similarity = 1 - (wed / wed_worst) if
wed_worst > 0 else 0.0
                total_similarity += similarity
                avg_similarity = (total_similarity /
self.num_tasks if self.num_tasks > 0 else 0.0).item()
                    reward['similarity'] += 50.0 * (2.0 *
avg_similarity - 1.0)
                std_workload =
torch.std(self.workloads[:self.num_employees]).item()
                max_std = self.max_workload / 2
                normalized_std = std_workload / max_std if
max_std > 0 else 0.0
                    reward['workload'] = 100.0 * (1.0 - 2.0 *
normalized_std)
                num_idle =
torch.sum(self.workloads[:self.num_employees] == 0).item()
                idle_ratio = num_idle / self.num_employees
                reward['idle'] += 50.0 * (1.0 - 2.0 * idle_ratio)
                info['avg_similarity'] = avg_similarity
                info['std_workload'] = std_workload
                info['num_idle'] = num_idle

        return self._get_obs(), reward, done, info

```

Lampiran 5. Kode DQN: Kelas jaringan saraf untuk DQN

```

class DQN(nn.Module):
    def __init__(self, max_employees, max_tasks,
num_employees, num_tasks):
        super(DQN, self).__init__()
        self.max_employees = max_employees
        self.max_tasks = max_tasks
        self.num_employees = num_employees
        self.num_tasks = num_tasks
        # Lapisan untuk memproses state_matrix,
workload_matrix, dan global_features
        self.fc_state = nn.Linear(max_tasks * 5,
128).to(device, dtype=torch.float32)
        self.fc_workload = nn.Linear(max_employees,
64).to(device, dtype=torch.float32)
        self.fc_global = nn.Linear(3, 16).to(device,
dtype=torch.float32)
        self.fc_combined = nn.Linear(128 + 64 + 16,
DQN_HIDDEN_DIM).to(device, dtype=torch.float32)
        self.fc2 = nn.Linear(DQN_HIDDEN_DIM, DQN_HIDDEN_DIM
// 2).to(device, dtype=torch.float32)
        self.fc3 = nn.Linear(DQN_HIDDEN_DIM // 2, 2 *
max_tasks * max_employees).to(device, dtype=torch.float32)

```

```

    def forward(self, state_matrix, workload_matrix,
global_features):
        batch_size = state_matrix.size(0)
        # Proses input melalui lapisan jaringan
        state_input = state_matrix.view(batch_size, -1)
        state_input = F.relu(self.fc_state(state_input))
        workload_input = workload_matrix.view(batch_size, -1)
        workload_input =
F.relu(self.fc_workload(workload_input))
        global_input = global_features.view(batch_size, -1)
        global_input = F.relu(self.fc_global(global_input))
        x = torch.cat([state_input, workload_input,
global_input], dim=1)
        x = F.relu(self.fc_combined(x))
        x = F.relu(self.fc2(x))
        q_values = self.fc3(x).view(batch_size, 2,
self.max_tasks, self.max_employees)
        # Mask tugas dan karyawan yang tidak valid
        q_values[:, :, :, self.num_tasks:, :] = -float('inf')
        q_values[:, :, :, :, self.num_employees:] =
-float('inf')
        return q_values

```

Lampiran 6. Kode DQN: Kelas agen untuk mengelola pelatihan DQN

```

class DQNAgent:
    def __init__(self, num_tasks, num_employees,
max_tasks=MAX_TASKS, max_employees=MAX_EMPLOYEES):
        self.device = device
        self.num_tasks = num_tasks
        self.num_employees = num_employees
        self.max_tasks = max_tasks
        self.max_employees = max_employees
        # Inisialisasi model dan target model
        self.model = DQN(self.max_employees, self.max_tasks,
self.num_employees, self.num_tasks).to(device)
        self.target_model = DQN(self.max_employees,
self.max_tasks, self.num_employees,
self.num_tasks).to(device)

        self.target_model.load_state_dict(self.model.state_dict())
        self.optimizer =
torch.optim.Adam(self.model.parameters(), lr=LEARNING_RATE)
        self.memory = deque(maxlen=MEMORY_SIZE)
        self.batch_size = BATCH_SIZE
        self.epsilon = INITIAL_EPSILON
        self.gamma = GAMMA
        self.global_step = 0
        self.scaler = torch.amp.GradScaler('cuda')

    # Memperbarui nilai epsilon untuk eksplorasi

```

```

def update_eps(self):
    self.global_step += 1
    self.epsilon = max(MIN_EPSILON, INITIAL_EPSILON *
np.exp(-DECAY_RATE_EPS * self.global_step))

    # Memilih aksi berdasarkan observasi
def act(self, obs, env):
    valid_tasks =
torch.where(obs['task_mask'][:self.num_tasks] == 1)[0]
    assigned_tasks =
torch.where(env.assignments[:self.num_tasks] != -1)[0]
    std_workload =
torch.std(env.workloads[:self.num_employees]).item()
    action_type = 0
    task_idx = 0
    emp_idx = 0
    if len(valid_tasks) == 0 and len(assigned_tasks) ==
0:
        return [action_type, task_idx, emp_idx]
    reassign_prob = 0.5 if (std_workload >
REASSIGNMENT_THRESHOLD or
torch.sum(env.workloads[:self.num_employees] == 0) > 0) else
0.1
    reassign_prob *= max(0.1, 1.0 - (env.total_reassigs /
300.0))
    if random.random() < self.epsilon:
        if len(valid_tasks) > 0:
            max_attempts = 10
            for _ in range(max_attempts):
                task_idx =
random.choice(valid_tasks.cpu().numpy())
                valid_emps = torch.where(
                    (env.workloads[:self.num_employees] +
env.story_points[task_idx] <= env.max_workload) &
((env.employee_projects[:self.num_employees] == -1) |
(env.employee_projects[:self.num_employees] ==
env.project_ids[task_idx]))
                )[0]
                if valid_emps.size(0) > 0:
                    emp_idx =
random.choice(valid_emps.cpu().numpy())
                    action_type = 0
                    break
            if action_type == 0 and len(valid_tasks) > 0 and
random.random() < reassign_prob:
                assignable_tasks = [t for t in
assigned_tasks.cpu().numpy() if env.reassign_counts[t] <
MAX_REASSIGN_PER_TASK]
                if len(assignable_tasks) > 0:
                    max_attempts = 10
                    for _ in range(max_attempts):
                        task_idx =

```

```

random.choice(assignable_tasks)
                                current_emp_idx =
env.assignments[task_idx].item()
                                valid_emps = torch.where(
                                    (torch.arange(self.num_employees,
device=device) != current_emp_idx) &
(env.workloads[:self.num_employees] +
env.story_points[task_idx] <= env.max_workload) &
((env.employee_projects[:self.num_employees] == -1) |
(env.employee_projects[:self.num_employees] ==
env.project_ids[task_idx]))
                                )[0]
                                if valid_emps.size(0) > 0:
                                    emp_idx =
random.choice(valid_emps.cpu().numpy())
                                    action_type = 1
                                    break
                                else:
                                    state_matrix =
obs['state_matrix'].unsqueeze(0).to(dtype=torch.float32)
                                    workload_matrix =
obs['workload_matrix'].unsqueeze(0).to(dtype=torch.float32)
                                    global_features =
obs['global_features'].unsqueeze(0).to(dtype=torch.float32)
                                    with torch.no_grad():
                                        q_values = self.model(state_matrix,
workload_matrix, global_features)
                                        task_mask = torch.zeros(self.max_tasks,
device=device, dtype=torch.bool)
                                        task_mask[valid_tasks] = 1
                                        emp_mask = torch.ones(self.max_tasks,
self.max_employees, device=device, dtype=torch.bool)
                                        for t in range(self.num_tasks):
                                            emp_mask[t, :self.num_employees] = (
                                                (env.workloads[:self.num_employees] +
env.story_points[t] <= env.max_workload) &
((env.employee_projects[:self.num_employees] == -1) |
(env.employee_projects[:self.num_employees] ==
env.project_ids[t]))
                                            )
                                            emp_mask[t, self.num_employees:] = False
                                            assign_mask = torch.zeros_like(q_values,
dtype=torch.bool)
                                            assign_mask[:, 0] =
task_mask.unsqueeze(-1).expand(-1, self.max_employees) &
emp_mask
                                            reassign_mask = torch.zeros_like(q_values,
dtype=torch.bool)
                                            for t in assigned_tasks:
                                                if env.reassign_counts[t] <

```

```

MAX_REASSIGN_PER_TASK:
    current_emp_idx =
env.assignments[t].item()
    reassign_emp_mask =
emp_mask[t].clone()
        reassign_emp_mask[current_emp_idx] =
False
            reassign_mask[0, 1, t] =
reassign_emp_mask
                combined_mask = assign_mask | reassign_mask
q_values[~combined_mask] = -float('inf')
if std_workload > REASSIGNMENT_THRESHOLD:
    q_values[:, 1, :, :] += 2.0
if torch.all(q_values == -float('inf')):
    if len(valid_tasks) > 0:
        max_attempts = 10
        for _ in range(max_attempts):
            task_idx =
random.choice(valid_tasks.cpu().numpy())
            valid_emps = torch.where(
(env.workloads[:self.num_employees] +
env.story_points[task_idx] <= env.max_workload) &
((env.employee_projects[:self.num_employees] == -1) |
(env.employee_projects[:self.num_employees] ==
env.project_ids[task_idx]))
) [0]
            if valid_emps.size(0) > 0:
                emp_idx =
random.choice(valid_emps.cpu().numpy())
                    action_type = 0
                    break
            elif len(assigned_tasks) > 0:
                assignable_tasks = [t for t in
assigned_tasks.cpu().numpy() if env.reassign_counts[t] <
MAX_REASSIGN_PER_TASK]
                if len(assignable_tasks) > 0:
                    max_attempts = 10
                    for _ in range(max_attempts):
                        task_idx =
random.choice(assignable_tasks)
                            current_emp_idx =
env.assignments[task_idx].item()
                            valid_emps = torch.where(
(torch.arange(self.num_employees, device=device) !=
current_emp_idx) &
(env.workloads[:self.num_employees] +
env.story_points[task_idx] <= env.max_workload) &
((env.employee_projects[:self.num_employees] == -1) |

```

```

(env.employee_projects[:self.num_employees] ==
env.project_ids[task_idx]))
) [0]
if valid_emps.size(0) > 0:
    emp_idx =
random.choice(valid_emps.cpu().numpy())
        action_type = 1
        break
    else:
        action_idx =
torch.argmax(q_values.view(-1)).item()
            action_type = action_idx //
(self.max_tasks * self.max_employees)
            task_emp_idx = action_idx %
(self.max_tasks * self.max_employees)
            task_idx = task_emp_idx //
self.max_employees
            emp_idx = task_emp_idx %
self.max_employees
        return [action_type, task_idx, emp_idx]

# Menyimpan pengalaman ke memori replay
def remember(self, state, action, reward, next_state,
done):
    self.memory.append((state, action, reward,
next_state, done))

# Melatih model menggunakan batch dari memori
def replay(self, env):
    if len(self.memory) < self.batch_size:
        return
    batch = random.sample(self.memory, self.batch_size)
    states, actions, rewards, next_states, dones =
zip(*batch)
    state_matrices = torch.stack([s['state_matrix'] for s
in states]).to(dtype=torch.float32)
    workload_matrices = torch.stack([s['workload_matrix'] for s
in states]).to(dtype=torch.float32)
    global_features = torch.stack([s['global_features'] for s
in states]).to(dtype=torch.float32)
    next_state_matrices = torch.stack([s['state_matrix'] for s
in next_states]).to(dtype=torch.float32)
    next_workload_matrices =
torch.stack([s['workload_matrix'] for s in
next_states]).to(dtype=torch.float32)
    next_global_features =
torch.stack([s['global_features'] for s in
next_states]).to(dtype=torch.float32)
    actions = torch.tensor(actions, dtype=torch.long,
device=self.device)
    rewards = torch.tensor([r['similarity'] +
r['workload'] + r['idle'] for r in rewards],
dtype=torch.float32, device=self.device)

```

```

dones = torch.tensor(dones, dtype=torch.float32,
device=self.device)
    action_types = actions[:, 0]
    task_idx = actions[:, 1]
    emp_idx = actions[:, 2]
    batch_indices = torch.arange(self.batch_size,
device=self.device)
    with torch.amp.autocast('cuda'):
        self.optimizer.zero_grad()
        q_values = self.model(state_matrices,
workload_matrices, global_features)
        q_values_selected = q_values[batch_indices,
action_types, task_idx, emp_idx]
        next_q_values =
self.target_model(next_state_matrices,
next_workload_matrices, next_global_features)
        next_q_values_max =
next_q_values.view(self.batch_size, -1).max(dim=1)[0]
        targets = rewards + (1 - dones) * self.gamma *
next_q_values_max
        loss = F.mse_loss(q_values_selected, targets)
        self.scaler.scale(loss).backward()
        self.scaler.step(self.optimizer)
        self.scaler.update()

# Memperbarui jaringan target
def update_target(self):

self.target_model.load_state_dict(self.model.state_dict())

# Menyimpan model dan parameter agen
def save(self, model_path, agent_path):
    torch.save({
        'model_state_dict': self.model.state_dict(),
        'target_model_state_dict':
self.target_model.state_dict(),
        'optimizer_state_dict':
self.optimizer.state_dict()
    }, model_path)
    torch.save({'epsilon': self.epsilon}, agent_path)

# Memuat model dan parameter agen
def load(self, model_path, agent_path):
    checkpoint = torch.load(model_path)

self.model.load_state_dict(checkpoint['model_state_dict'])

self.target_model.load_state_dict(checkpoint['target_model_st
ate_dict'])

self.optimizer.load_state_dict(checkpoint['optimizer_state_di
ct'])
    agent_params = torch.load(agent_path)

```

```
    self.epsilon = agent_params['epsilon']
    self.memory = deque(maxlen=MEMORY_SIZE)
```

Lampiran 7. Fungsi untuk memvisualisasikan & merangkum hasil pelatihan

```
def plot_and_summarize(episode_data, wed_scores_list,
                      total_rewards, similarity_rewards,
                      workload_balance_rewards, idle_rewards,
                      workloads_list,
                      std_workloads_list,
                      idle_employees_list, env, num_episodes_to_plot,
                      checkpoint_episode=None):
    print(f"\nRangkuman Hasil Pelatihan hingga Episode {num_episodes_to_plot}:")
    print("-" * 80)
    print(f"{'Episode':<10} {'Total Reward':<15} "
          f"{'Similarity':<15} {'Workload':<15} {'Idle':<15} {'Std Dev':<10} {'Idle Emp':<10}")
    print("-" * 80)
    for data in episode_data:
        if data['Episode'] <= num_episodes_to_plot:
            print(f"{data['Episode']:<10} {data['Total Reward']:<15.4f} "
                  f"{data['Similarity Reward']:<15.4f} "
                  f"{data['Workload Balance Reward']:<15.4f} {data['Idle Reward']:<15.4f} {data['Std Workloads']:<10.4f} {data['Idle Employees']:<10}")
    print("-" * 80)

    plt.figure(figsize=(15, 20))
    plt.subplot(4, 2, 1)
    alpha = ALPHA_WED
    weights_worst = 1 / (1 + alpha * 1)
    wed_raw_worst = np.sqrt(np.sum(weights_worst * (0 - 1) ** 2) * env.num_skills)
    standardized_similarities_per_episode = []
    for episode_wed_scores in wed_scores_list[:num_episodes_to_plot]:
        similarities = [(wed_raw_worst - wed) / wed_raw_worst
                        for wed in episode_wed_scores]
    standardized_similarities_per_episode.append(similarities)
    plt.boxplot(standardized_similarities_per_episode,
                tick_labels=[f"Ep {i+1}" for i in range(num_episodes_to_plot)])
    plt.title(f'Distribusi Skor Kesamaan (Hingga Episode {num_episodes_to_plot})')
    plt.xlabel('Episode')
    plt.ylabel('Skor Kesamaan (0 hingga 1)')
    plt.grid(True, linestyle='--', alpha=0.7)

    plt.subplot(4, 2, 2)
    plt.plot(range(1, num_episodes_to_plot + 1),
             np.array(total_rewards[:num_episodes_to_plot]), marker='o',
```

```

label='Total Reward')
    plt.title(f'Total Reward (Hingga Episode
{num_episodes_to_plot})')
    plt.xlabel('Episode')
    plt.ylabel('Total Reward')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.legend()

    plt.subplot(4, 2, 3)
    plt.plot(range(1, num_episodes_to_plot + 1),
np.array(similarity_rewards[:num_episodes_to_plot]),
marker='o', label='Similarity Reward', color='blue')
    plt.plot(range(1, num_episodes_to_plot + 1),
np.array(workload_balance_rewards[:num_episodes_to_plot]),
marker='o', label='Workload Balance Reward', color='green')
    plt.plot(range(1, num_episodes_to_plot + 1),
np.array(idle_rewards[:num_episodes_to_plot]), marker='o',
label='Idle Reward', color='red')
    plt.title(f'Reward per Tujuan (Hingga Episode
{num_episodes_to_plot})')
    plt.xlabel('Episode')
    plt.ylabel('Total Reward')
    plt.grid(True, linestyle='--', alpha=0.7)
    plt.legend()

    plt.subplot(4, 2, 4)
    workloads = np.array(workloads_list[num_episodes_to_plot
- 1][:env.num_employees])
    plt.bar(range(env.num_employees), workloads)
    plt.title(f'Distribusi Story Points (Episode
{num_episodes_to_plot})')
    plt.xlabel('Indeks Karyawan')
    plt.ylabel('Total Story Points')

    plt.subplot(4, 2, 5)
    plt.bar(range(1, num_episodes_to_plot + 1),
np.array(std_workloads_list[:num_episodes_to_plot]))
    plt.title(f'Deviasi Standar Beban Kerja (Hingga Episode
{num_episodes_to_plot})')
    plt.xlabel('Episode')
    plt.ylabel('Deviasi Standar')

    plt.subplot(4, 2, 6)
    plt.bar(range(1, num_episodes_to_plot + 1),
np.array(idle_employees_list[:num_episodes_to_plot]))
    plt.title(f'Jumlah Karyawan Idle (Hingga Episode
{num_episodes_to_plot})')
    plt.xlabel('Episode')
    plt.ylabel('Karyawan Idle')
    plt.ylim(bottom=0)

plt.tight_layout()
filename =

```

```

f'reassignment_finalcode_training_metrics_eps{checkpoint_episode}.png' if checkpoint_episode else
'reassignment_finalcode_training_metrics_final.png'
plt.savefig(filename)
print(f"Plot metrik pelatihan disimpan ke '{filename}'")
plt.show()

```

Lampiran 8. Fungsi utama untuk melatih agen DQN

```

# Fungsi utama untuk melatih agen DQN
def train_dqn(task_data_path, employee_data_path):
    # Memuat data tugas dan karyawan
    task_data = pd.read_csv(task_data_path)
    employee_data = pd.read_csv(employee_data_path)
    env = TaskAssignmentEnv(task_data, employee_data)
    agent = DQNAgent(env.num_tasks, env.num_employees)
    total_rewards = []
    wed_scores_list = []
    workloads_list = []
    std_workloads_list = []
    idle_employees_list = []
    similarity_rewards = []
    workload_balance_rewards = []
    idle_rewards = []
    global_step = 0
    episode_data = []
    action_distribution = []

    # Iterasi untuk setiap episode pelatihan
    for episode in range(NUM_EPISODES):
        obs = env.reset()
        total_reward = 0
        episode_similarity = 0
        episode_workload_balance = 0
        episode_idle_reward = 0
        invalid_actions = 0
        done = False
        step = 0
        action_counts = {'assign': 0, 'reassign': 0,
        'emp_assignments': torch.zeros(env.num_employees)}
        print(f"Episode {episode+1}/{NUM_EPISODES}, Epsilon: {agent.epsilon:.3f}")
        while not done:
            step += 1
            global_step += 1
            start_time = time.time()
            action = agent.act(obs, env)
            next_obs, reward, done, info = env.step(action)
            agent.remember(obs, action, reward, next_obs, done)
            agent.replay(env)
            obs = next_obs
            total_reward += sum(reward.values())
            episode_similarity += reward['similarity']

```

```

        episode_workload_balance += reward['workload']
        episode_idle_reward += reward['idle']
        action_type, _, emp_idx = action
        if action_type == 0:
            action_counts['assign'] += 1
        else:
            action_counts['reassign'] += 1
        action_counts['emp_assignments'][emp_idx] += 1
        if not info.get('action_valid', True):
            invalid_actions += 1
        num_idle =
            torch.sum(env.workloads[:env.num_employees] == 0).item()
        if step % 10 == 0:
            step_time = time.time() - start_time
            print(f"Step {step}: Reward
{total_reward/step:.4f}, Time {step_time:.2f}s, Num Idle:
{num_idle}")
            agent.update_eps()
            if global_step % TARGET_UPDATE_FREQ == 0:
                agent.update_target()

        wed_scores_list.append(env.wed_scores)

workloads_list.append(env.workloads[:env.num_employees].clone().c
pu().numpy())
        std_workloads = info.get('std_workload',
torch.std(env.workloads[:env.num_employees]).item())
        num_idle = info.get('num_idle',
torch.sum(env.workloads[:self.num_employees] == 0).item())
        std_workloads_list.append(std_workloads)
        idle_employees_list.append(max(num_idle, 0))
        total_rewards.append(total_reward)
        similarity_rewards.append(episode_similarity)
        workload_balance_rewards.append(episode_workload_balance)
        idle_rewards.append(episode_idle_reward)
        print(f"Episode {episode+1} - Steps: {step}, Total
Reward: {total_reward:.2f}, Std Workloads: {std_workloads:.2f},
Idle: {num_idle}, Invalid Actions: {invalid_actions}")
        print(f"Total Similarity Reward:
{episode_similarity:.2f}, Workload Balance Reward:
{episode_workload_balance:.2f}, Idle Reward:
{episode_idle_reward:.2f}")
        episode_data.append({
            'Episode': episode + 1,
            'Total Reward': total_reward,
            'Similarity Reward': episode_similarity,
            'Workload Balance Reward': episode_workload_balance,
            'Idle Reward': episode_idle_reward,
            'Std Workloads': std_workloads,
            'Idle Employees': max(num_idle, 0)
        })
        action_dist_data = {
            'Episode': episode + 1,
            'Assign_Count': action_counts['assign'],
            'Reassign_Count': action_counts['reassign'],

```

```

        'Emp_Assignments':
action_counts['emp_assignments'].cpu().numpy().tolist()
    }
    action_distribution.append(action_dist_data)
    action_dist_df = pd.DataFrame(action_distribution)
    action_dist_df.to_csv(os.path.join(log_dir,
'finalcode_action_distribution.csv'), index=False)

    if episode + 1 in SAVE_EPISODES:
        print(f"\nCheckpoint pada Episode {episode + 1}")
        plot_and_summarize(
            episode_data=episode_data,
            wed_scores_list=wed_scores_list,
            total_rewards=total_rewards,
            similarity_rewards=similarity_rewards,
            workload_balance_rewards=workload_balance_rewards,
            idle_rewards=idle_rewards,
            workloads_list=workloads_list,
            std_workloads_list=std_workloads_list,
            idle_employees_list=idle_employees_list,
            env=env,
            num_episodes_to_plot=episode + 1,
            checkpoint_episode=episode + 1
        )
        checkpoint_model_path =
f"/content/drive/MyDrive/Skripsi/reassignment_finalcode_dqn_model
_eps{episode + 1}.pth"
        checkpoint_agent_path =
f"/content/drive/MyDrive/Skripsi/reassignment_finalcode_dqn_agent
_params_eps{episode + 1}.pth"
        agent.save(checkpoint_model_path,
checkpoint_agent_path)

        agent.save(model_save_path, agent_save_path)
plot_and_summarize(
    episode_data=episode_data,
    wed_scores_list=wed_scores_list,
    total_rewards=total_rewards,
    similarity_rewards=similarity_rewards,
    workload_balance_rewards=workload_balance_rewards,
    idle_rewards=idle_rewards,
    workloads_list=workloads_list,
    std_workloads_list=std_workloads_list,
    idle_employees_list=idle_employees_list,
    env=env,
    num_episodes_to_plot=NUM_EPISODES
)
    return env, workloads_list, std_workloads_list,
idle_employees_list, NUM_EPISODES, episode_data, agent

```

Lampiran 9. Fungsi untuk menjalankan pelatihan

```
# Menjalankan pelatihan
env, workloads_list, std_workloads_list, idle_employees_list,
num_episodes, episode_data, trained_agent =
train_dqn(task_data_path, employee_data_path)
```

Lampiran 10. Link Github Keseluruhan Kode

<https://github.com/wisnuwdn/Task-Optimization-DQN>