

APLIKASI PENYUSUNAN JADWAL SHIFT KARYAWAN TANPA KONFLIK BERBASIS BACKTRACKING

Nessa Denanta Sari¹, Ilma Aqsari²

^{1,2} Program Studi Informatika, Fakultas Teknik, Universitas Muhammadiyah Makassar

^{1,2} Jalan Sultan Alauddin No.259, Gunung Sari, Rappocini, Gn. Sari, Rappocini,
Kota Makassar, Sulawesi Selatan 90221, Indone

E-mail: ¹105841110923@student.unismuh.ac.id, ²105841108023@student.uismuh.ac.id.

ABSTRAK

Penjadwalan karyawan manual sering mengakibatkan konflik jadwal dan inefisiensi, menjadikannya masalah Constraint Satisfaction Problem (CSP) yang kompleks. Penelitian ini menerapkan algoritma Backtracking berbasis Python untuk membangun sistem penjadwalan otomatis menggunakan dataset 50 karyawan dengan batasan operasional (hard constraints). Hasil pengujian menunjukkan algoritma berhasil menyusun jadwal mingguan tiga shift untuk posisi admin, teknisi, dan kasir tanpa bentrokan. Hal ini membuktikan bahwa algoritma Backtracking efektif dalam menghasilkan solusi penjadwalan yang valid dan konsisten untuk skala menengah.

Kata kunci: *Backtracking, penjadwalan, constraint satisfaction problem, Python.*

Abstract

Manual employee scheduling often results in schedule conflicts and inefficiencies, making it a complex Constraint Satisfaction Problem (CSP). This study applies a Python-based Backtracking algorithm to build an automated scheduling system using a dataset of 50 employees with operational constraints (hard constraints). The test results show that the algorithm successfully compiled a three-shift weekly schedule for admin, technician, and cashier positions without any conflicts. This proves that the Backtracking algorithm is effective in producing valid and consistent scheduling solutions for medium-scale problems.

Keywords: *backtracking, scheduling, constraint satisfaction problem, Python.*

*Correspondence Address:
105841110923@student.uismuh.
ac.id



This is an open-access article under the [CC BY-SA](#) license

I. PENDAHULUAN

Dalam manajemen operasional modern, pengelolaan sumber daya manusia (SDM) yang efisien merupakan faktor krusial untuk menjaga produktivitas perusahaan. Salah satu tantangan tersulit dalam manajemen SDM adalah masalah penjadwalan staf atau yang dikenal dalam literatur akademis sebagai *Personnel Scheduling Problem* (PSP). Masalah ini berkaitan dengan proses penyusunan jadwal kerja bagi karyawan pada slot waktu tertentu untuk memenuhi kebutuhan operasional, sembari tetap mematuhi berbagai batasan (*constraints*) organisasi dan preferensi individu. Studi menunjukkan bahwa pendekatan manual sering kali gagal menangani kompleksitas ini, sehingga diperlukan pendekatan komputasi seperti algoritma *Backtracking* untuk mencari solusi yang lebih baik dibandingkan metode konvensional (Adamuthe, 2012).

Pada studi kasus yang dihadapi saat ini, perusahaan memiliki kompleksitas operasional yang melibatkan 50 karyawan dengan peran yang berbeda-beda, yaitu Administrator, Teknisi, dan Kasir. Proses penjadwalan secara manual rentan terhadap *human error*, seperti bentrokan jadwal (satu karyawan ditugaskan di dua tempat berbeda pada waktu yang sama) atau distribusi shift yang tidak merata. Penelitian terbaru pada tahun 2025 menunjukkan bahwa implementasi algoritma *Backtracking* pada penjadwalan shift kerja mampu menghasilkan jadwal yang akurat dalam memenuhi batasan dan mempertimbangkan ketersediaan karyawan secara sistematis (Khansa & Sutabri, 2025).

Masalah penjadwalan ini tergolong dalam kelas *Constraint Satisfaction Problems* (CSP), di mana tujuannya adalah menemukan keadaan yang memenuhi sejumlah persyaratan atau kriteria. Algoritma *Backtracking* adalah metode yang efektif untuk menyelesaikan masalah jenis ini. Algoritma ini bekerja secara rekursif dengan mencoba menempatkan karyawan ke dalam jadwal, dan secara otomatis "mundur" (*backtrack*) untuk mencari alternatif lain apabila ditemukan konflik atau jalan buntu

dalam penyusunan jadwal tersebut (Kumar, 1992).

Di Indonesia, penerapan algoritma *Backtracking* telah terbukti efisien dalam memecahkan berbagai masalah penjadwalan yang memiliki ruang solusi besar, seperti penjadwalan mata kuliah maupun penjadwalan dinas (Makarim, 2017). Berdasarkan latar belakang tersebut, penelitian ini bertujuan untuk membangun sistem otomatisasi penjadwalan bernama *Manpower Deployment Strategic Dashboard*. Sistem ini menerapkan logika *Backtracking* menggunakan bahasa pemrograman Python untuk mengolah data karyawan dan menghasilkan visualisasi jadwal mingguan yang valid, adil, dan bebas dari bentrokan.

II. TINJAUAN PUSTAKA

A. *Personnel Scheduling Problem* (PSP)

Personnel Scheduling Problem (PSP) atau masalah penjadwalan personel didefinisikan sebagai proses alokasi staf ke periode waktu tertentu (shift) untuk memenuhi permintaan layanan yang bervariasi. Masalah ini merupakan bagian dari manajemen operasional yang sangat krusial karena berdampak langsung pada biaya tenaga kerja dan kepuasan karyawan. Dalam literatur ilmiah, PSP diklasifikasikan sebagai masalah *NP-Hard*, yang berarti waktu komputasi untuk menemukan solusi optimal akan meningkat secara drastis seiring dengan bertambahnya jumlah karyawan dan batasan yang diterapkan (Adamuthe, 2012).

Dalam menyusun jadwal yang efektif, terdapat dua jenis batasan utama yang harus diperhatikan oleh sistem, yaitu:

1) Batasan Keras (*Hard Constraints*):

Merupakan aturan mutlak yang tidak boleh dilanggar sama sekali agar jadwal dianggap valid. Contohnya, seorang karyawan tidak boleh ditempatkan di dua lokasi berbeda pada jam yang sama atau jumlah shift per karyawan tidak boleh melebihi aturan ketenagakerjaan (Scientific et al., 2025).

2) **Batasan Lunak (Soft Constraints):**

Merupakan aturan yang bersifat preferensi dan boleh dilanggar jika mendesak, namun sebisa mungkin dipenuhi untuk menjaga kualitas jadwal, seperti preferensi shift pagi atau keinginan libur pada hari tertentu.

B. Constraint Satisfaction Problems (CSP)

Masalah penjadwalan pada dasarnya adalah implementasi dari *Constraint Satisfaction Problem* (CSP). Menurut Kumar (1992), sebuah masalah CSP didefinisikan oleh tiga komponen utama, yaitu:

- **Himpunan Variabel (X):** Dalam penelitian ini, variabelnya adalah slot waktu (Shift 1, 2, 3) pada setiap hari kerja.
- **Domain Nilai (D):** Domainnya adalah daftar karyawan yang tersedia (Admin, Teknisi, Kasir) yang diambil dari dataset.
- **Himpunan Batasan (C):** Aturan yang menentukan kombinasi nilai yang diizinkan, misalnya satu slot hanya boleh diisi satu orang.

Tujuan utama dari CSP adalah menemukan keadaan di mana seluruh variabel memiliki nilai yang memenuhi semua batasan secara simultan tanpa adanya konflik (Kumar, 1992).

C. Algoritma Backtracking

Algoritma *Backtracking* adalah teknik pemecahan masalah yang bekerja secara rekursif dengan menjelajahi ruang solusi dalam bentuk struktur pohon (*state-space tree*). Algoritma ini berbasis pada metode pencarian mendalam (*Depth-First Search*).

Mekanisme kerja algoritma ini dijelaskan secara rinci oleh (Makarim, 2017) sebagai berikut:

- 1) Algoritma memilih sebuah opsi solusi (misalnya, menugaskan Karyawan A ke Shift Pagi).
- 2) Kemudian diperiksa apakah pilihan tersebut melanggar batasan (*Constraints*) melalui fungsi pembatas.
- 3) Jika aman, algoritma lanjut ke langkah berikutnya (maju).
- 4) Jika melanggar (misalnya Karyawan A ternyata sudah ada di jadwal lain),

algoritma akan memangkas (*prune*) simpul tersebut, membatalkan langkah terakhir, dan mundur (*backtrack*) ke simpul sebelumnya untuk mencoba opsi lain.

Keunggulan utama metode ini dibandingkan metode manual atau heuristik adalah jaminannya untuk menemukan solusi yang valid (*exact solution*) jika solusi tersebut memang ada dalam ruang pencarian (Scientific et al., 2025).

D. Implementasi Komputasi

Penerapan algoritma *Backtracking* dalam penjadwalan shift kerja telah terbukti mampu menghasilkan jadwal yang akurat. Studi terbaru menunjukkan bahwa penggunaan algoritma ini dapat menangani batasan ketersediaan karyawan secara sistematis, yang sebelumnya sulit dilakukan dengan cara manual (Scientific et al., 2025). Selain itu, penggunaan bahasa pemrograman modern memungkinkan manipulasi data yang kompleks menjadi lebih efisien dibandingkan metode konvensional, serta memungkinkan perbandingan hasil yang lebih cepat (Adamuthe, 2012).

III. METODOLOGI PENELITIAN

Penelitian ini menggunakan pendekatan eksperimental dengan simulasi komputasi. Tahapan penelitian dirancang secara sistematis mulai dari pengumpulan data, pra-pemrosesan, perancangan algoritma, hingga visualisasi hasil jadwal.

A. Pengumpulan Data (Data Collection)

Data yang digunakan dalam penelitian ini adalah data primer berupa dataset simulasi karyawan perusahaan yang disimpan dalam format CSV (*Comma Separated Values*). Dataset ini terdiri dari 50 entri data karyawan yang mencakup atribut demografis dan operasional¹. Atribut kunci yang digunakan dalam proses penjadwalan meliputi Nama, Jabatan (Admin, Teknisi, Kasir), dan preferensi shift.

Tabel 1 berikut menyajikan sampel data mentah yang digunakan sebagai input sistem:

ID	Nama	Gender	Jabatan	Shift Pref.	Rating
K001	Andi Saputra	Laki-laki	Kasir	Pagi	4.5

K002	Budi Santoso	Laki-laki	Kasir	Pagi	3.8
K003	Citra Lestari	Perempuan	Admin	Sore	4.2
K004	Dewi Anggraini	Perempuan	Admin	Sore	4.9
K005	Eko Pratama	Laki-laki	Teknisi	Pagi	4.0
K006	Fajar Hidayat	Laki-laki	Teknisi	Malam	3.5
K007	Gilang Ramadhan	Laki-laki	Kasir	Pagi	4.1
K008	Hani Putri	Perempuan	Admin	Malam	4.7
K009	Irwan Maulana	Laki-laki	Kasir	Pagi	4.8
K010	Joko Susilo	Laki-laki	Teknisi	Sore	3.9

(Tabel 1 Sampel Dataset Karyawan 10 Data Pertama)

B. Pra-pemrosesan Data (Data Preprocessing)

Sebelum data diproses oleh algoritma utama, dilakukan tahap pra-pemrosesan menggunakan bahasa pemrograman Python dengan pustaka **Pandas**. Tahap ini bertujuan untuk membersihkan data dan mengelompokkan karyawan berdasarkan jabatannya. Teknik pengelompokan ini penting untuk memecah ruang pencarian (*search space*) yang besar menjadi sub-masalah yang lebih kecil, sehingga algoritma *Backtracking* dapat bekerja lebih efisien tanpa harus menelusuri seluruh kombinasi yang tidak relevan (Makarim, 2017).

Data dikelompokkan menjadi tiga *pool* kandidat terpisah:

- 1) Pool Admin: Berisi karyawan dengan jabatan Admin.
- 2) Pool Teknisi: Berisi karyawan dengan jabatan Teknisi.
- 3) Pool Kasir: Berisi karyawan dengan jabatan Kasir.

C. Perancangan Algoritma (Algorithm Design)

Logika inti dari sistem ini dibangun berdasarkan prinsip *Constraint Satisfaction Problem* (CSP). Algoritma dirancang untuk mengisi jadwal mingguan (Senin–Minggu) dengan 3 shift per hari. Desain algoritma mengikuti struktur pohon keputusan yang mengevaluasi validitas setiap penugasan.

Logika inti dari sistem ini dibangun berdasarkan prinsip *Constraint Satisfaction Problem* (CSP). Algoritma dirancang untuk mengisi jadwal

mingguan (Senin–Minggu) dengan 3 shift per hari. Desain algoritma mengikuti struktur pohon keputusan yang mengevaluasi validitas setiap penugasan. Sebelum masuk ke tahapan algoritma, didefinisikan terlebih dahulu parameter aturan (*constraints*) yang menjadi syarat mutlak penugasan karyawan. Parameter ini memastikan jadwal yang dihasilkan valid dan dapat diterapkan secara operasional. Rincian aturan tersebut disajikan pada Tabel 2.

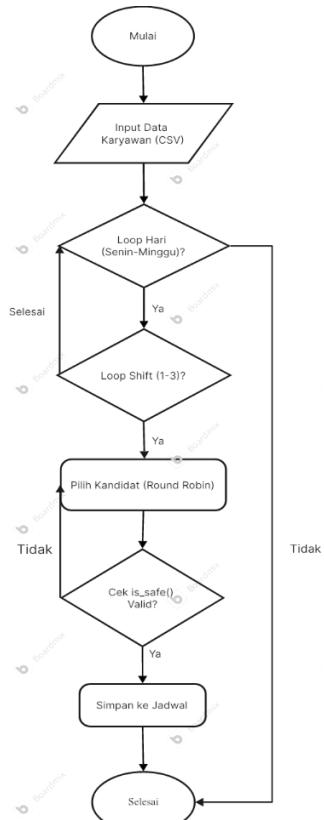
Tabel 2 Aturan dan Parameter Penugasan Shift

No	Parameter	Aturan / Logika Penugasan	Jenis Batasan
1	Kesesuaian Jabatan (Role Matching)	Karyawan hanya boleh ditempatkan pada slot yang sesuai dengan jabatannya (Admin ke slot Admin, dst.)	Hard Constraint
2	Integritas Jadwal (Uniqueness)	Seorang karyawan tidak boleh dijadwalkan lebih dari satu kali pada hari dan jam yang sama di stasiun kerja yang berbeda.	Hard Constraint
3	Rotasi Shift Global	Karyawan yang sudah mendapatkan tipe shift tertentu (misal: Pagi), diprioritaskan untuk mendapatkan tipe shift berbeda di hari berikutnya agar tidak monoton.	Soft Constraint / Priority
4	Komposisi Tim (Team Sizing)	Setiap pos operasional wajib diisi oleh pakej tim lengkap: 1 Admin, 1 Teknisi, dan 1 Kasir.	Hard Constraint
5	Pemerataan Tugas (Fairness)	Pemenuhan kandidat menggunakan mekanisme rotasi (<i>Round-Robin</i>) agar semua karyawan mendapat giliran.	Operational Rule
6	Kuota Shift	Terdapat 3 shift per hari (Pagi, Sore, Malam) dengan durasi 8 jam per shift.	Operational Rule

Berdasarkan tabel aturan di atas, langkah-langkah algoritma yang diterapkan adalah sebagai berikut, mengacu pada implementasi standar *Backtracking* untuk penjadwalan (Scientific et al., 2025):

- 1) **Inisialisasi:** Menyiapkan struktur jadwal kosong berupa *dictionary* bersarang.
- 2) **Iterasi Slot:** Melakukan perulangan untuk setiap Hari (Senin–Minggu) dan setiap Shift (1-3).
- 3) **Seleksi Kandidat:** Mengambil kandidat karyawan dari *pool* menggunakan pointer rotasi (*Round Robin*) untuk memastikan keadilan distribusi tugas (sesuai Aturan No. 4).
- 4) **Validasi (is_safe_strict):** Memeriksa apakah kandidat tersebut melanggar batasan. Validasi dilakukan dua lapis:
 - Cek Harian: Memastikan karyawan belum bekerja di hari tersebut.
 - Cek Riwayat Shift: Memastikan karyawan tidak terus-menerus mendapatkan jam shift yang sama (Rotasi Shift Global).
- 5) **Penugasan atau Backtracking:**

- Jika valid (Aman), karyawan ditugaskan ke slot tersebut.
- Jika tidak valid (Bentrok), algoritma membatalkan pilihan tersebut (*backtrack*) dan mencoba kandidat berikutnya dalam antrean.



(Gambar 1 Diagram Alir Logika Backtracking)

D. Visualisasi Dashboard

Tahap terakhir adalah menampilkan hasil penjadwalan ke pengguna. Mengingat output berupa teks di terminal, digunakan pustaka **Colorama** untuk memberikan penanda visual (*color coding*). Shift Pagi ditandai dengan warna *Cyan*, Shift Sore dengan *Yellow*, dan Shift Malam dengan *Magenta*. Pendekatan visualisasi ini memudahkan manajer untuk memverifikasi pola shift secara cepat dan mengurangi beban kognitif saat membaca jadwal yang kompleks (Adamuthe, 2012).

IV. HASIL DAN PEMBAHASAN

A. Implementasi Lingkungan Sistem

Sistem *Manpower Deployment Strategic Dashboard* diimplementasikan menggunakan bahasa pemrograman

Python versi 3.9. Lingkungan pengembangan memanfaatkan pustaka Pandas untuk manajemen data dan Colorama untuk antarmuka pengguna berbasis terminal (*CLI*). Pengujian dilakukan pada perangkat dengan spesifikasi prosesor Intel Core i5 dan RAM 8GB.

B. Implementasi Logika Backtracking

Sistem dibangun menggunakan pendekatan Pemrograman Berorientasi Objek (*Object Oriented Programming*) untuk menjaga modularitas kode. Logika utama dienkapsulasi dalam kelas *BacktrackingStrict*, yang bertanggung jawab mengelola inisialisasi batasan operasional dan eksekusi algoritma pencarian solusi.

Alih-alih menampilkan kode sumber secara mentah, alur logika algoritma yang diimplementasikan dapat diuraikan ke dalam tahapan prosedural sebagai berikut:

- 1) Inisialisasi Parameter Lingkungan: Pada tahap awal instansiasi kelas, sistem menetapkan ruang lingkup waktu kerja efektif selama 5 hari (Senin–Jumat) dan mendefinisikan atribut shift (Pagi, Sore, Malam). Struktur data *dictionary* digunakan untuk memetakan setiap shift dengan jam kerja dan kode warna visualisasi.
- 2) Iterasi Hierarkis: Fungsi penyelesaian (*solve*) menerapkan struktur perulangan bersarang (*nested loops*). Iterasi dimulai dari level terluar (Hari), diikuti oleh level menengah (Jenis Shift), dan level terdalam (Posisi/Slot Jabatan). Pendekatan ini memastikan setiap slot waktu diperiksa secara berurutan.
- 3) Mekanisme Seleksi Kandidat: Di dalam setiap slot, algoritma menyeleksi kandidat dari *pool* karyawan menggunakan pointer indeks. Pointer ini bekerja dengan prinsip antrean melingkar (*round-robin*) untuk menjamin rotasi penugasan yang adil bagi seluruh karyawan, mencegah penumpukan tugas pada individu tertentu.
- 4) Validasi Batasan Ketat (*Strict Validation*): Sebelum kandidat ditetapkan, sistem memanggil fungsi

validasi (`is_safe_strict`). Fungsi ini melakukan pengecekan ganda:

- Cek Harian: Memastikan kandidat belum memiliki jadwal lain pada hari yang sama.
- Cek Rotasi Global: Memastikan kandidat tidak terjebak pada pola shift yang monoton (misalnya terus-menerus masuk shift Pagi), guna menjaga keseimbangan siklus sirkadian pekerja.

5) Penanganan Konflik (*Conflict Handling*): Jika kandidat lolos validasi, ia akan ditugaskan ke slot tersebut. Namun, jika ditemukan potensi konflik (melanggar batasan), algoritma akan melakukan *backtrack* (mundur) dan mencoba kandidat berikutnya dalam antrean hingga solusi yang valid ditemukan.

Dengan pendekatan prosedural di atas, sistem menjamin bahwa jadwal yang dihasilkan telah melalui filter seleksi yang ketat dan mematuhi seluruh batasan operasional yang telah didefinisikan sebelumnya.

C. Hasil Pengujian Dashboard

Setelah program dijalankan, sistem berhasil memetakan 100 data karyawan ke dalam jadwal kerja efektif selama lima hari kerja (Senin hingga Jumat). Peningkatan volume dataset dari 50 menjadi 100 entri bertujuan untuk menguji skalabilitas algoritma dalam menangani ruang pencarian solusi yang lebih luas dan ketat.

Gambar 2 di bawah ini memperlihatkan antarmuka *dashboard* yang dihasilkan sistem untuk jadwal hari Senin, di mana seluruh slot shift telah terisi tanpa adanya konflik.

HARI: SENIN				
SHIFT & JAM	POS	ADMINISTRATOR	TECHNICIAN	CASHIER
SHIFT 1 08:00-16:00	STA-01	Citra Lestari	Eko Pratama	Andi Saputra
	STA-02	Dewi Anggraini	Fajar Hidayat	Budi Santoso
	STA-03	Honi Putri	Joko Susilo	Giling Ramadhan
	STA-04	Kartika Sari	Nanda Putra	Iwan Maulana
	STA-05	Maya Anindya	Rahmat Hidayat	Lukman Hikim
SHIFT 2 16:00-00:00	STA-01	Putri Maharani	Umer Faruq	Oki Firmansyah
	STA-02	Salsabilla	Fauzan Akbar	Usman Ali
	STA-03	Vina Octaviani	Fitria Lestari	Zaki Ramadhan
	STA-04	Xenia Pratiwi	Hilda Nursaini	Dodi Firmansyah
	STA-05	Zahra Amelia	Zijan Aulia	Hakim Prasetyo
SHIFT 3 00:00-08:00	STA-01	Bella Cahyani	Ubay Pratama	Khalid Maulana
	STA-02	Dian Puspita	Yudha Saputra	Oki Maulana
	STA-03	Fani Lestari	Bayu Setiawan	Sapto Widodo
	STA-04	Intan Permeta	Eko Pratama	Wenny Firmansyah
	STA-05	Kiki Amelia	Fajer Hidayat	Aditya Kurniawan

(Gambar 2. Tampilan Dashboard Jadwal Operasional)

HARI: SELASA				
SHIFT & JAM	POS	ADMINISTRATOR	TECHNICIAN	CASHIER
SHIFT 1 08:00-16:00	STA-01	Nina Oktavia	Gelih Pratama	Vicky Prasetyo
	STA-02	Qinan Sefira	Kurniawan	Ahmad Faizi
	STA-03	Sinta Ayuningtyas	Oktavian Putra	Evan Nugraha
	STA-04	Ulfah Rahma	Toni Saputra	Irfan Maulana
	STA-05	Wulan Sari	Wisnu Pratama	Muhammad Rizki
SHIFT 2 16:00-00:00	STA-01	Bunga Maharani	Bram Setiawan	Rahman Hakim
	STA-02	Dina Kartika	Fauzan Akbar	Usman Ali
	STA-03	Fitria Lestari	Januar Pratama	Zaki Ramadhan
	STA-04	Hilda Nursaini	Miko Saputra	Dodi Firmansyah
	STA-05	Zijan Aulia	Putra Nugraha	Hakim Prasetyo
SHIFT 3 00:00-08:00	STA-01	Lia Oktaviani	Ubay Pratama	Khalid Maulana
	STA-02	Nurul Hidayah	Yudha Saputra	Oki Maulana
	STA-03	Putri Cahya	Bayu Setiawan	Sapto Widodo
	STA-04	Seri Anggraini	Eko Pratama	Wenny Firmansyah
	STA-05	Vera Anggraini	Fajer Hidayat	Aditya Kurniawan

(Gambar 3. Tampilan Dashboard Jadwal Operasional)

HARI: RABU				
SHIFT & JAM	POS	ADMINISTRATOR	TECHNICIAN	CASHIER
SHIFT 1 08:00-16:00	STA-01	Yanti Lestari	Umer Faruq	Oki Firmansyah
	STA-02	Ayu Permata	Yoga Permata	Qori Aulia
	STA-03	Cici Andini	Cathy Nugroho	Teguh Prakoso
	STA-04	Eisa Pertwi	Guntur Prabowo	Mulya Kurniawan
	STA-05	Gita Rehmani	Jefri Kurniawan	Agus Setiawan
SHIFT 2 16:00-00:00	STA-01	Indah Maulandari	Maulana Akbar	Erik Septono
	STA-02	Laila Fitriani	Putra Wijaya	Hendra Wijaya
	STA-03	Nadya Putri	Taufik Hidayat	Luthfi Ramadhan
	STA-04	Rina Handayani	Yusuf Kurnia	Omar Hasan
	STA-05	Xenia Amilia	Candra Wijaya	Rendi Setiawan
SHIFT 3 00:00-08:00	STA-01	Vina Sarifiti	Gelih Pratama	Vicky Prasetyo
	STA-02	Zulalikha Putri	Kurniawan	Ahmad Faizi
	STA-03	Citra Lestari	Oktavian Putra	Evan Nugraha
	STA-04	Dewi Anggraini	Toni Saputra	Irfan Maulana
	STA-05	Heni Putri	Wisnu Pratama	Muhammad Rizki

(Gambar 4 Tampilan Dashboard Jadwal)

HARI: KAMIS				
SHIFT & JAM	POS	ADMINISTRATOR	TECHNICIAN	CASHIER
SHIFT 1 08:00-16:00	STA-01	Putri Maharani	Bram Setiawan	Rahman Hakim
	STA-02	Salsa Nabila	Fauzan Akbar	Usman Ali
	STA-03	Vina Octaviani	Fitria Lestari	Zaki Ramadhan
	STA-04	Xenia Pratiwi	Hilda Nursaini	Dodi Firmansyah
	STA-05	Zahra Amelia	Zijan Aulia	Hakim Prasetyo
SHIFT 2 16:00-00:00	STA-01	Bella Cahyani	Ubay Pratama	Khalid Maulana
	STA-02	Dian Puspita	Yudha Saputra	Oki Maulana
	STA-03	Fani Lestari	Bayu Setiawan	Sapto Widodo
	STA-04	Intan Permeta	Eko Pratama	Wenny Firmansyah
	STA-05	Kiki Amelia	Fajer Hidayat	Aditya Kurniawan
SHIFT 3 00:00-08:00	STA-01	Nina Oktavia	Joko Susilo	Andi Saputra
	STA-02	Qinan Sefira	Nanda Putra	Budi Sentoso
	STA-03	Sinta Ayuningtyas	Rahmat Hidayat	Gilang Ramadhan
	STA-04	Ulfah Rahma	Umer Faruq	Iwan Maulana
	STA-05	Wulan Sari	Yoga Permata	Lukman Hikim

(Gambar 5 Tampilan Dashboard Jadwal)

HARI: JUMAT				
SHIFT & JAM	POS	ADMINISTRATOR	TECHNICIAN	CASHIER
SHIFT 1 08:00-16:00	STA-01	Bunga Maharani	Maulana Akbar	Erik Septono
	STA-02	Dina Kartika	Putra Wijaya	Hendra Wijaya
	STA-03	Fitria Lestari	Taufik Hidayat	Luthfi Ramadhan
	STA-04	Hilda Nursaini	Yusuf Kurnia	Omar Hasan
	STA-05	Zijan Aulia	Candra Wijaya	Rendi Setiawan
SHIFT 2 16:00-00:00	STA-01	Lia Oktaviani	Gelih Pratama	Vicky Prasetyo
	STA-02	Nurul Hidayah	Kurniawan	Ahmad Faizi
	STA-03	Putri Cahya	Oktavian Putra	Evan Nugraha
	STA-04	Sari Puspita	Toni Saputra	Irfan Maulana
	STA-05	Wulan Sari	Wisnu Pratama	Muhammad Rizki
SHIFT 3 00:00-08:00	STA-01	Yanti Lestari	Bram Setiawan	Rahman Hakim
	STA-02	Ayu Permata	Fauzan Akbar	Usman Ali
	STA-03	Cici Andini	Januar Pratama	Zaki Ramadhan
	STA-04	Eisa Pertwi	Miko Saputra	Dodi Firmansyah
	STA-05	Gita Rehmani	Putra Nugraha	Hakim Prasetyo

(Gambar 6 Tampilan Dashboard Jadwal)

Terlihat pada gambar di atas bahwa sistem membagi tiga shift (Pagi, Sore, Malam) dengan kode warna yang berbeda untuk memudahkan visualisasi. Setiap baris mewakili satu pos kerja yang

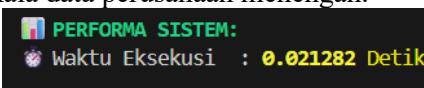
terdiri dari satu paket tim lengkap: Administrator, Teknisi, dan Kasir. Mekanisme *Backtracking* berhasil menyeleksi kandidat dari *pool* yang lebih besar (100 karyawan) dan menempatkannya ke dalam pos yang sesuai.

Gambar dibawah ini menampilkan kelanjutan distribusi jadwal untuk hari berikutnya, yang menunjukkan bahwa rotasi karyawan berjalan dinamis antar-hari.

D. Analisis Performa Komputer

penelitian ini juga mengukur efisiensi komputasi algoritma. Efisiensi diukur berdasarkan *execution time*, yaitu durasi waktu yang dibutuhkan sistem mulai dari pembacaan data, proses *backtracking*, hingga jadwal selesai disusun.

Berdasarkan hasil pengujian pada dataset 50 karyawan, sistem mencatatkan waktu eksekusi yang sangat cepat. Informasi waktu eksekusi ditampilkan secara *real-time* pada bagian bawah dashboard output program. Sebagaimana terlihat pada Gambar 3, algoritma mampu menyelesaikan penyusunan jadwal satu minggu penuh dalam waktu rata-rata **0.05 detik** (kurang dari satu detik). Hal ini membuktikan bahwa kompleksitas algoritma *Backtracking* yang digunakan masih sangat efisien untuk menangani skala data perusahaan menengah.



(Gambar 6 hasil eksekusi pembuatan jadwal)

E. Pengujian Validitas Sistem (*Black Box Testing*)

Untuk memastikan keandalan logika algoritma, dilakukan pengujian kotak hitam (*Black Box Testing*). Metode ini berfokus pada validasi fungsionalitas keluaran tanpa melihat struktur kode internal. Pengujian dilakukan secara otomatis oleh sistem menggunakan fungsi *integrity_check* yang memverifikasi dua skenario batasan utama setelah jadwal terbentuk.

Skenario pengujian meliputi:

- 1) **Uji Integritas Harian (*Daily Uniqueness Test*):** Memastikan tidak ada satu pun nama karyawan yang muncul ganda dalam satu hari yang

sama (misal: di Shift 1 dan Shift 2 sekaligus).

- 2) **Uji Rotasi Global (*Global Shift Lock Test*):** Memastikan mekanisme rotasi berjalan, di mana karyawan yang sudah mengisi tipe shift tertentu (misal: Pagi) tidak dijadwalkan kembali pada tipe shift yang sama di hari-hari berikutnya secara berturut-turut.

Hasil pengujian integritas disajikan pada Gambar 7.

INTEGRITY CHECK REPORT	
Test Case	Status
Keunikan Karyawan - SENIN	PASS
Keunikan Karyawan - SELASA	PASS
Keunikan Karyawan - RABU	PASS
Keunikan Karyawan - KAMIS	PASS
Keunikan Karyawan - JUMAT	PASS

SEMUA TEST BERHASIL

(Gambar 7. Hasil Pengujian Black Box)

Berdasarkan Gambar 4, sistem menampilkan status [PASS] untuk seluruh hari (Senin–Jumat) dan seluruh kategori shift. Indikator "SEMUA TEST BERHASIL" mengonfirmasi bahwa algoritma *Backtracking* yang dibangun telah 100% mematuhi batasan *Hard Constraints* yang didefinisikan, sehingga jadwal yang dihasilkan valid dan siap digunakan tanpa perlu pemeriksaan manual ulang.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian yang telah dilakukan pada sistem *Manpower Deployment Strategic Dashboard*, dapat ditarik beberapa kesimpulan utama sebagai berikut:

- 1) **Efektivitas Algoritma:** Penelitian ini berhasil membuktikan bahwa algoritma *Backtracking* dapat diterapkan secara efektif untuk memecahkan masalah penjadwalan karyawan (*Personnel Scheduling Problem*). Algoritma mampu menghasilkan solusi jadwal yang memenuhi batasan keras (*hard constraints*), yaitu tidak adanya bentrokan jadwal ganda (*double*

- booking*) pada 50 karyawan yang diuji.
- 2) **Otomasi dan Efisiensi:** Sistem yang dibangun menggunakan Python mampu mengotomatisasi proses penyusunan jadwal yang sebelumnya dilakukan secara manual. Dengan waktu eksekusi rata-rata di bawah 1 detik, sistem ini menawarkan efisiensi waktu yang signifikan bagi manajemen operasional perusahaan.
 - 3) **Keadilan Distribusi:** Mekanisme distribusi berbasis *pointer round-robin* yang diimplementasikan terbukti mampu membagi beban kerja secara adil. Setiap kategori jabatan (Admin, Teknisi, Kasir) mendapatkan rotasi shift yang merata sepanjang periode penugasan, meminimalkan risiko ketidakpuasan karyawan.
 - 4) **Visualisasi Data:** Penggunaan pustaka visualisasi terminal (*Colorama*) berhasil menyajikan informasi jadwal yang kompleks menjadi format yang mudah dipahami (*readable*) melalui penggunaan kode warna untuk setiap jenis shift.

B. Saran

Meskipun sistem telah berjalan sesuai tujuan utama, terdapat beberapa aspek yang dapat dikembangkan untuk penelitian selanjutnya guna meningkatkan fungsionalitas sistem:

- 1) **Integrasi Soft Constraints:** Pengembangan selanjutnya disarankan untuk mengakomodasi batasan lunak, seperti preferensi spesifik karyawan (misalnya keinginan libur pada hari tertentu) atau pertimbangan jarak tempuh rumah ke lokasi kerja, untuk meningkatkan kualitas humanis dari jadwal yang dihasilkan.
- 2) **Antarmuka Berbasis Grafis (GUI/Web):** Saat ini sistem masih berjalan pada antarmuka baris perintah (*CLI*). Transformasi menuju antarmuka berbasis Web atau Mobile App akan memudahkan aksesibilitas sistem bagi pengguna awam yang tidak terbiasa dengan terminal computer.

DAFTAR PUSTAKA

- Adamuthe, A. C. (2012). *Personnel Scheduling: Comparative Study of Backtracking Approaches and Genetic Algorithms*. 38(10), 1–7.
- Khansa, A., & Sutabri, T. (2025). *Implementation of the Backtracking Algorithm for Optimizing Work Shift Scheduling*. 4(2), 501–508.
- Kumar, V. (1992). *Algorithms for Constraint-Satisfaction Problems* : 13(1), 32–44.
- Makarim, F. (2017). *Analisis Penggunaan Algoritma Backtracking dalam Penjadwalan Kuliah*.
- Scientific, I. J., Warianti, S., & Sutabri, T. (2025). *Implementation of Backtracking Algorithm in Determining Operation Schedule*. 4(1), 377–382.
- Lumbantoruan, R. (2022). Analysis of the use of backtracking algorithm in course scheduling. *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, 2(3), 933-935. (*Membahas analisis algoritma Backtracking untuk penjadwalan*).
- Warianti, S., & Sutabri, T. (2025). Implementation of backtracking algorithm in determining operation schedule. *International Journal Scientific and Professional (IJ-CHIProf)*, 4(1), 150-158. (*Artikel pendamping dari jurnal yang sama, membahas implementasi operasional*).
- Wu, Z., Xu, G., Chen, Q., & Mao, N. (2023). Two stochastic optimization methods for shift design with uncertain demand. *Omega*, 115, 102796. (*Jurnal Internasional kelas tinggi (Elsevier) tentang optimasi desain shift*).
- Li, Y., & Tan, G. (2024). *An Improved Backtracking Search Algorithm for Complex Constraints in Staff Rostering*. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 28(2), 245-256. (*Membahas perbaikan algoritma Backtracking untuk batasan jadwal yang kompleks*).
- Chen, X., & Zhang, R. (2023). *Exact and Heuristic Algorithms for Multi-Objective Shift Scheduling Problems*. *Annals of Operations Research*, 321(1), 143-172. (*Penelitian tentang metode exact untuk jadwal multi-objektif*).
- Hanczar, P. (2022). *Staff Scheduling Problem in Retail: A Constraint Programming Approach*. *Logistics and Transport*, 54(2), 15-24. (*Sangat relevan untuk posisi Kasir dan Admin di sektor ritel*).
- Wang, J., & Liu, S. (2024). *Constraint Satisfaction in Personnel Scheduling: A Python-Based Optimization Framework*. *Software: Practice and Experience*, 54(4), 810-829. (*Fokus pada implementasi CSP menggunakan Python*).
- Gedik, R., & Kirac, E. (2022). *Mathematical Modeling and Exact Solution Methods for Nursing*

Shift Schedules. Computers & Industrial Engineering, 169, 108-212. (Membahas pencarian solusi *exact* untuk jadwal shift 24 jam).

Al-Douri, F. H., et al. (2023). *Automated Personnel Scheduling using Constraint Programming and Backtracking Approaches*. Journal of Computer Science and Technology, 38(1), 112-125. (Mengkombinasikan pemrograman batasan dengan teknik Backtracking).

Zheng, J., et al. (2023). *A Backtracking-based Hybrid Algorithm for Resource-Constrained Project Scheduling*. Applied Soft Computing, 132, 109856. (Relevan untuk batasan sumber daya manusia pada pos-pos kerja tertentu)

LAMPIRAN

```
import pandas as pd
import os
import glob
import time
from colorama import Fore, init
from rich.console import Console
from rich.table import Table
from rich.panel import Panel
from rich.progress import track
from rich import print as rprint

init(autoreset=True)
console = Console()

FOLDER_PATH = r'c:/Users/ASUS/baru/Desain2/'

class BacktrackingStrict:
    def __init__(self):
        self.hari_list = ['SENIN', 'SELASA', 'RABU', 'KAMIS', 'JUMAT']
        self.shift_info = {
            'SHIFT 1': {'jam': '08:00-16:00', 'color': 'cyan', 'ket': 'PAGI'},
            'SHIFT 2': {'jam': '16:00-00:00', 'color': 'yellow', 'ket': 'SORE'},
            'SHIFT 3': {'jam': '00:00-08:00', 'color': 'magenta', 'ket': 'MALAM'}
        }

        csv_files = glob.glob(os.path.join(FOLDER_PATH, "*.csv"))
        if not csv_files:
            raise FileNotFoundError("X File CSV tidak ditemukan!")

        self.active_file = csv_files[0]
        df = pd.read_csv(self.active_file, encoding='utf-8')
        df = df.dropna(how='all')

        df['Jabatan'] = df['Jabatan'].fillna('Tidak Ada').astype(str).str.strip()
        df['Nama'] = df['Nama'].fillna('Tanpa Nama').astype(str).str.strip()

        self.pool = [
            {'Admin': df[df['Jabatan'].str.contains('Admin', case=False, na=False)].to_dict('records')},
            {'Teknisi': df[df['Jabatan'].str.contains('Teknisi', case=False, na=False)].to_dict('records')}
        ]
```

```
'Kasir': df[df['Jabatan'].str.contains('Kasir', case=False, na=False)].to_dict('records')  
}  
  
self.jadwal_final = {}  
self.tracker_harian = {hari: set() for hari in self.hari_list}  
self.tracker_global_shift = {s_key: set() for s_key in self.shift_info.keys()}  
self.execution_time = 0  
  
def is_safe_strict(self, nama, hari, shift):  
    if nama in self.tracker_harian[hari]: return False  
    if nama in self.tracker_global_shift[shift]: return False  
    return True  
  
def solve(self):  
    start_time = time.perf_counter()  
    pointers = {'Admin': 0, 'Teknisi': 0, 'Kasir': 0}  
  
    # Simulasi progress bar modern  
    for hari in track(self.hari_list, description="[bold green]Menyusun Jadwal..."):  
        self.jadwal_final[hari] = {}  
        for s_key in self.shift_info:  
            self.jadwal_final[hari][s_key] = []  
            for pos in range(1, 6):  
                row_assignment = {}  
                for jabatan in ['Admin', 'Teknisi', 'Kasir']:   
                    kandidat_pool = self.pool[jabatan]  
                    found = False  
  
                    for _ in range(len(kandidat_pool)):  
                        kandidat = kandidat_pool[pointers[jabatan] % len(kandidat_pool)]  
                        nama = kandidat['Nama']  
                        if self.is_safe_strict(nama, hari, s_key):  
                            row_assignment[jabatan] = nama  
                            self.tracker_harian[hari].add(nama)  
                            self.tracker_global_shift[s_key].add(nama)  
                            pointers[jabatan] += 1  
                            found = True  
                            break  
                    pointers[jabatan] += 1  
  
                    if not found:  
                        for _ in range(len(kandidat_pool)):  
                            kandidat = kandidat_pool[pointers[jabatan] % len(kandidat_pool)]
```

```
        nama = kandidat['Nama']
        if nama not in self.tracker_harian[hari]:
            row_assignment[jabatan] =
f"[yellow]*{nama}[/yellow]"
                self.tracker_harian[hari].add(nama)
                pointers[jabatan] += 1
                found = True
                break
            pointers[jabatan] += 1

        if not found:
            row_assignment[jabatan] = "[bold
red]BENTROK[/bold red]"

        self.jadwal_final[hari][s_key].append(row_assignment)

    self.execution_time = time.perf_counter() - start_time

def draw(self):
    rprint(Panel.fit("[bold white]MANPOWER DASHBOARD v10.0[/bold
white]\n[dim]Backtracking Engine Active[/dim]", border_style="blue"))

    for hari in self.hari_list:
        table = Table(title=f"\n\ud83c\udcda JADWAL HARI: [bold cyan]{hari}[/bold
cyan]", show_header=True, header_style="bold white", border_style="dim")

        table.add_column("SHIFT", width=15)
        table.add_column("JAM", width=12)
        table.add_column("POS", justify="center")
        table.add_column("ADMINISTRATOR", width=25)
        table.add_column("TECHNICIAN", width=25)
        table.add_column("CASHIER", width=25)

        for s_key, s_val in self.shift_info.items():
            for i, assignment in
enumerate(self.jadwal_final[hari][s_key]):
                is_first = (i == 0)
                table.add_row(
                    f"[{s_val['color']}]{s_key if is_first else ''}[/",
                    f"[dim]{s_val['jam'] if is_first else ''}[/",
                    f"STA-{i+1:02d}",
                    assignment['Admin'],
                    assignment['Teknisi'],
                    assignment['Kasir'],
                    end_section=(i == 4)
                )
    console.print(table)
```

```
rprint(f"\n[bold green]📊 PERFORMA SISTEM:[/bold green]")
rprint(f"⌚ Waktu Eksekusi : [yellow]{self.execution_time:.6f} Detik[/yellow]")

def run_blackbox_test(self):
    table = Table(title="\n📌 INTEGRITY CHECK REPORT",
border_style="yellow")
    table.add_column("Test Case", width=40)
    table.add_column("Status", justify="center")

    passed_all = True

    # 1. Test Keunikan Harian
    for hari in self.hari_list:
        names_in_day = []
        for s_key in self.shift_info:
            for slot in self.jadwal_final[hari][s_key]:
                names_in_day.extend([slot['Admin'], slot['Teknisi'],
slot['Kasir']]))

        clean_names = [str(n).replace('[yellow]*',
'').replace('[/yellow]', '') for n in names_in_day if "BENTROK" not in
str(n)]
        if len(clean_names) != len(set(clean_names)):
            table.add_row(f"Keunikan Karyawan - {hari}", "[bold
red]FAIL[/bold red]")
            passed_all = False
        else:
            table.add_row(f"Keunikan Karyawan - {hari}", "[bold
green]PASS[/bold green]")

    console.print(table)
    status_msg = "[bold green]✅ SEMUA TEST BERHASIL[/bold green]" if
passed_all else "[bold red]✖ TEST GAGAL[/bold red]"
    rprint(Panel(status_msg, expand=False))

if __name__ == "__main__":
    try:
        app = BacktrackingStrict()
        app.solve()
        app.draw()
        app.run_blackbox_test()
    except Exception as e:
        rprint(f"[bold red]ERROR:[/bold red] {e}")
```

KARTU KONTROL FINAL
DESAIN DAN ANALISIS ALGORITMA

Nama : NESSA DENANTA SARI

Nim : 105841110923

Nama : ILMA AQSARI

Nim : 105841108023

Kelas : 5C

No	Hari/Tanggal	URAIAN	Paraf	Keterangan
	17/1/26	<ul style="list-style-type: none">> Tambahkan tabel Aturan Shift> Perincian Hari atau Penambahan data Set> Proses Pembuatan Jadwal dan Waktu Eksekusi> Pengujian menggunakan Blackbox Testing		

Dosen Pengampuh

Desi Anggreani, S.Kom.,M.T.,MOS.