JITET (Jurnal Informatika dan Teknik Elektro Terapan)
Vol. 14 No. 3, pISSN: 2303-0577 eISSN: 2830-7062 http://dx.doi.org/10.23960/jitet.v14i3.10078

PERBANDINGAN ALGORITMA GREEDY, BACKTRACKING, DAN BRUTE FORCE PADA PENJADWALAN INDIAN PREMIER LEAGUE

Wahyudi Alfurqon[1], Muhammad Tri Setianto[2], Ahmad Naufal[3], Muhammad Ezar Al Rivan[4]

[1,2,3,4]Program Studi Informatika, Universitas Multi Data Palembang; Jalan Rajawali Nomor 14, Palembang, Sumatera Selatan, 30113

Keywords:
Combinatorial optimization, Constraint satisfaction, Greedy algorithm, IPL scheduling, backtracking.

Corespondent Email:
[1]wahyudialfurqon_2327250069@mhs.mdp.ac.id

Abstrak. Penjadwalan pertandingan olahraga profesional seperti *Indian Premier League* merupakan permasalahan kombinatorial yang kompleks karena melibatkan keterbatasan waktu, tempat pertandingan, dan konflik antar tim. Penelitian ini bertujuan membandingkan kinerja algoritma *greedy*, *backtracking*, dan *brute force* dalam menyelesaikan masalah penjadwalan IPL. Data yang digunakan adalah dataset jadwal IPL 2022 yang terdiri atas 70 pertandingan, sepuluh tim, dan lima tempat pertandingan. Penelitian dilakukan dengan menguji ketiga algoritma pada enam fase jumlah pertandingan, yaitu 15, 20, 25, 30, 35, dan 70 pertandingan, dengan masing-masing sepuluh kali percobaan. Parameter evaluasi meliputi waktu eksekusi, jumlah pertandingan valid, dan jumlah konflik jadwal. Hasil penelitian menunjukkan bahwa algoritma *greedy* memiliki waktu eksekusi rata-rata 0,0001 detik dengan kemampuan menangani 70 pertandingan, sementara algoritma *backtracking* memerlukan waktu 1.643 detik pada fase 30 pertandingan. Algoritma *brute force* menunjukkan waktu eksekusi 13.947 detik pada fase 30 pertandingan. Ketiga algoritma berhasil menghasilkan jadwal tanpa konflik. Kesimpulan penelitian ini adalah algoritma *greedy* lebih unggul dalam efisiensi waktu dan skalabilitas untuk penjadwalan IPL skala besar.

Abstract. *Scheduling professional sports matches such as the Indian Premier League is a complex combinatorial problem due to time constraints, venue limitations, and team conflicts. This study aims to compare the performance of greedy, backtracking, and brute force algorithms in solving the IPL scheduling problem. The dataset used is the IPL 2022 schedule consisting of 70 matches, ten teams, and five venues. The research was conducted by testing the three algorithms on six phases of match counts, namely 15, 20, 25, 30, 35, and 70 matches, with ten trials each. Evaluation parameters include execution time, number of valid matches, and number of scheduling conflicts. The results show that the greedy algorithm has an average execution time of 0.0001 seconds and can handle 70 matches, while the backtracking algorithm requires 1,643 seconds at the 30-match phase. The brute force algorithm shows an execution time of 13,947 seconds at the 30-match phase. All three algorithms successfully produced conflict-free schedules. This study concludes that the greedy algorithm is superior in time efficiency and scalability for large-scale IPL scheduling.*

1. PENDAHULUAN

Industri olahraga profesional, khususnya turnamen kriket *Indian Premier League* (IPL), memiliki kompleksitas penjadwalan yang tinggi. Turnamen IPL mempertemukan sepuluh tim dalam satu musim dengan total 70 pertandingan yang harus dijadwalkan dalam kurun waktu kurang lebih dua bulan. Setiap pertandingan memiliki atribut berupa tanggal, waktu, tim tuan rumah, tim tamu, dan tempat pertandingan. Tantangan utama dalam penjadwalan IPL adalah memastikan tidak terjadi bentrokan jadwal, baik dari sisi tim maupun tempat pertandingan.

Permasalahan penjadwalan termasuk ke dalam kelas masalah kombinatorial yang kompleks [1]. Jumlah kemungkinan jadwal yang dapat dibentuk dari 70 pertandingan dengan berbagai batasan sangatlah besar, sehingga diperlukan pendekatan algoritmik yang efisien. Penjadwalan yang tidak tepat dapat mengakibatkan tim harus bermain dua kali pada hari yang sama atau sebuah tempat pertandingan harus menyelenggarakan dua pertandingan secara bersamaan, yang secara operasional tidak memungkinkan.

Penjadwalan ulang atau restorasi jadwal yang mengalami gangguan juga menjadi isu penting dalam operasional turnamen [2]. Ketika terjadi waktu henti (*off-time*) atau pembatalan pertandingan karena cuaca atau faktor lainnya, diperlukan algoritma yang mampu menyusun ulang jadwal dengan cepat dan tetap memenuhi seluruh batasan yang ada.

Berbagai pendekatan algoritmik telah dikembangkan untuk menyelesaikan masalah penjadwalan. Algoritma *greedy* menawarkan solusi cepat dengan membuat keputusan optimal pada setiap langkah tanpa mempertimbangkan konsekuensi di masa depan [3]. Sementara itu, algoritma *backtracking* melakukan eksplorasi solusi secara sistematis dengan mekanisme runut-balik ketika menemui jalan buntu [4]. Algoritma *brute force* menjadi pembanding dasar yang mengeksplorasi seluruh kemungkinan solusi yang ada [5].

Beberapa penelitian sebelumnya telah melakukan perbandingan algoritma untuk berbagai kasus optimasi. Penelitian tentang perbandingan algoritma *greedy* dan *backtracking* pada masalah pewarnaan graf menunjukkan bahwa algoritma *greedy* memiliki kompleksitas waktu yang lebih baik dibandingkan *backtracking* [6]. Penelitian lain tentang optimasi rute pengiriman barang juga menegaskan keunggulan algoritma *greedy* dari sisi kecepatan komputasi [7]. Namun, penelitian tentang perbandingan ketiga algoritma pada kasus penjadwalan pertandingan olahraga skala besar seperti IPL masih terbatas.

Berdasarkan uraian tersebut, penelitian ini bertujuan untuk membandingkan kinerja algoritma *greedy*, *backtracking*, dan *brute force* dalam menyelesaikan masalah penjadwalan pertandingan IPL. Hasil penelitian diharapkan dapat memberikan rekomendasi pemilihan algoritma yang tepat untuk sistem penjadwalan pertandingan olahraga skala besar.

2. TINJAUAN PUSTAKA

2.1. Algoritma Greedy

Algoritma *greedy* merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Kata *greedy* berarti rakus atau tamak, yang mencerminkan prinsip algoritma ini dalam mengambil keputusan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi di masa depan [8]. Algoritma ini membentuk solusi langkah demi langkah, di mana setiap langkah dipilih solusi yang paling optimal pada saat itu. Algoritma *greedy* telah diterapkan dalam berbagai masalah optimasi seperti pencarian jalur terpendek [9], penjadwalan [10], dan optimalisasi penjadwalan mesin tunggal [11].

2.2. Algoritma Backtracking

Algoritma *backtracking* atau runut-balik adalah algoritma yang berbasis pada *Depth First Search* (DFS) untuk mencari solusi persoalan secara lebih sistematis [9]. Algoritma ini merupakan perbaikan dari algoritma *brute force* karena tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan. Prinsip kerja algoritma ini adalah membangun pohon solusi secara rekursif dan melakukan runut-balik ketika menemui jalan buntu.

2.3. Algoritma Brute Force

Algoritma *brute force* adalah pendekatan yang sangat jelas dan sederhana dalam memecahkan suatu masalah, biasanya didasarkan pada pernyataan masalah dan definisi konsep yang dilibatkan [10]. Algoritma

ini memecahkan masalah dengan cara yang sangat sederhana, langsung, dan eksplisit. Seluruh kemungkinan solusi dieksplorasi secara lengkap, sehingga algoritma ini menjamin ditemukannya solusi optimal namun dengan konsekuensi waktu komputasi yang sangat besar.

2.4. Kompleksitas Waktu

Kompleksitas waktu adalah ukuran seberapa cepat waktu eksekusi suatu algoritma bertambah seiring dengan bertambahnya ukuran masukan. Notasi *Big-O* digunakan untuk menyatakan batas atas dari laju pertumbuhan suatu fungsi [12], [13]. Algoritma dengan kompleksitas polinomial seperti O (n^2) dianggap efisien, sementara algoritma dengan kompleksitas eksponensial seperti O (2^n) hanya cocok untuk ukuran data yang kecil.

2.5. Penjadwalan IPL

Indian Premier League (IPL) adalah liga kriket profesional di India yang diikuti oleh sepuluh tim. Setiap musim terdiri atas 70 pertandingan yang berlangsung selama kurang lebih dua bulan. Penjadwalan IPL harus memperhatikan berbagai batasan, seperti tidak ada tim yang bermain dua kali pada hari yang sama dan tidak ada dua pertandingan yang menggunakan tempat yang sama pada waktu yang bersamaan [14].

3. METODE PENELITIAN

3.1. Deskripsi Dataset

Dataset yang digunakan dalam penelitian ini adalah jadwal resmi IPL musim 2022 yang diperoleh dari platform Kaggle. Dataset terdiri atas 70 pertandingan dengan atribut sebagaimana ditampilkan pada Tabel 1.

Tabel 1. Struktur Dataset IPL 2022

Atribut	Tipe Data	Contoh Nilai
DATE	String	26/03/22
TIME	String	7:30
PM/AM	String	PM
HOME TEAM	String	Chennai Super Kings
AWAY TEAM	String	Kolkata Knight Riders
VENUE	String	Wankhede Stadium

Berdasarkan hasil eksplorasi data, terdapat sepuluh tim yang berpartisipasi dalam IPL 2022. Kesepuluh tim tersebut adalah Chennai Super Kings, Delhi Capitals, Gujarat Titans, Kolkata Knight Riders, Lucknow Super Giants, Mumbai Indians, Punjab Kings, Rajasthan Royals, Royal Challengers Bangalore, dan Sunrisers Hyderabad.

Seluruh pertandingan berlangsung pada periode 26 Maret 2022 hingga 22 Mei 2022 dengan menggunakan lima tempat pertandingan utama, yaitu Wankhede Stadium, Brabourne CCI, DY Patil Stadium, MCA Stadium Pune, dan beberapa tempat pertandingan pendukung lainnya. Dataset dinyatakan bersih dan siap digunakan karena tidak ditemukan nilai kosong pada seluruh atribut.

3.2. Formulasi Masalah Penjadwalan

Masalah penjadwalan pertandingan IPL diformulasikan ke dalam model *Constraint Satisfaction Problem* (CSP) dengan ketentuan sebagai berikut.

Variabel keputusan dalam penelitian ini adalah setiap pertandingan yang tercatat dalam dataset. Setiap pertandingan memiliki atribut tanggal, waktu, tim tuan rumah, tim tamu, dan tempat pertandingan. Solusi penjadwalan adalah himpunan bagian dari seluruh pertandingan yang dapat dijadwalkan tanpa melanggar batasan yang telah ditetapkan.

Batasan yang digunakan dalam penelitian ini terdiri atas dua aturan utama. Pertama, pada waktu yang sama, tidak boleh terdapat dua pertandingan yang menggunakan tempat pertandingan yang sama. Kedua, pada waktu yang sama, sebuah tim hanya boleh bertanding satu kali. Kedua batasan ini mencerminkan kondisi operasional nyata dalam penyelenggaraan turnamen olahraga profesional.

Fungsi objektif dari penelitian ini adalah memaksimalkan jumlah pertandingan yang dapat dijadwalkan secara valid. Suatu jadwal dinyatakan valid jika seluruh pertandingan dalam jadwal tersebut memenuhi kedua batasan yang telah ditetapkan.

3.3. Implementasi Algoritma Greedy

Algoritma *greedy* diimplementasikan dengan pendekatan seleksi berurutan. Seluruh pertandingan dalam dataset diacak terlebih dahulu untuk menghindari bias urutan data. Algoritma kemudian memproses setiap pertandingan satu per satu berdasarkan urutan hasil pengacakan.

Untuk setiap pertandingan yang sedang diproses, algoritma memeriksa apakah pertandingan tersebut dapat ditambahkan ke dalam jadwal yang sudah terbentuk. Pemeriksaan dilakukan dengan mengevaluasi apakah slot waktu yang digunakan oleh pertandingan tersebut, yang didefinisikan oleh tanggal dan jam pertandingan, masih tersedia. Suatu slot waktu dinyatakan tersedia jika pada slot yang sama belum terdapat pertandingan yang menggunakan tempat pertandingan yang sama atau melibatkan tim yang sama.

Jika pertandingan memenuhi kriteria kelayakan, pertandingan tersebut langsung ditambahkan ke dalam jadwal. Slot waktu yang bersangkutan kemudian ditandai sebagai telah terpakai. Proses ini berlanjut hingga seluruh pertandingan dalam dataset telah diproses. Kompleksitas waktu algoritma *greedy* dalam implementasi ini dinyatakan pada persamaan (1).

$$T(n) = O(n^2) \quad (1)$$

dengan n adalah jumlah pertandingan yang akan dijadwalkan. Kompleksitas ini berasal dari proses perbandingan antara setiap pertandingan dengan seluruh pertandingan yang telah masuk ke dalam jadwal untuk memastikan tidak terjadi konflik.

3.4. Implementasi Algoritma Backtracking

Algoritma *backtracking* diimplementasikan dengan pendekatan rekursif. Setiap pertandingan dalam dataset diproses secara berurutan dengan dua pilihan keputusan, yaitu mengambil pertandingan tersebut ke dalam jadwal jika memenuhi batasan atau melewatkannya.

Algoritma membangun pohon solusi secara rekursif. Pada setiap tingkat rekursi yang mewakili satu pertandingan, algoritma pertama-tama memeriksa apakah pertandingan tersebut dapat dimasukkan ke dalam jadwal sementara. Pemeriksaan dilakukan dengan membandingkan tanggal, waktu, tempat pertandingan, dan keterlibatan tim dengan seluruh pertandingan yang telah masuk ke dalam jadwal sementara.

Jika pertandingan dapat dimasukkan, algoritma akan melanjutkan rekursi ke tingkat berikutnya dengan jadwal yang telah diperbarui. Setelah eksplorasi pada cabang tersebut selesai, algoritma akan mengeluarkan pertandingan tersebut dari jadwal (backtrack) dan mencoba cabang alternatif dengan melewatkan pertandingan tersebut.

Algoritma *backtracking* terus melakukan eksplorasi hingga seluruh kemungkinan kombinasi telah dievaluasi. Solusi terbaik yang ditemukan selama proses pencarian, yaitu jadwal dengan jumlah pertandingan terbanyak, disimpan sebagai keluaran akhir. Kompleksitas waktu algoritma *backtracking* dalam kasus terburuk dinyatakan pada persamaan (2).

$$T(n) = O(2^n) \quad (2)$$

Dengan n adalah jumlah pertandingan yang akan dijadwalkan. Setiap penambahan satu pertandingan menyebabkan ruang pencarian solusi menjadi dua kali lipat lebih besar, yang berdampak langsung pada peningkatan waktu eksekusi secara eksponensial.

3.5. Implementasi Algoritma Brute Force

Algoritma *brute force* diimplementasikan dengan mengeksplorasi seluruh kemungkinan himpunan bagian dari kumpulan pertandingan. Algoritma membangkitkan seluruh kombinasi pertandingan mulai dari ukuran satu hingga ukuran maksimum menggunakan fungsi kombinasi.

Setiap kombinasi yang terbentuk diperiksa kelayakannya dengan memastikan bahwa seluruh pertandingan dalam kombinasi tersebut dapat dijadwalkan secara simultan tanpa melanggar batasan. Kombinasi terbaik yang ditemukan, yaitu kombinasi dengan jumlah pertandingan terbanyak dan tidak mengandung konflik, disimpan sebagai solusi. Kompleksitas waktu algoritma *brute force* dinyatakan pada persamaan (3).

$$T(n) = O(2^n \times n^2) \quad (3)$$

dengan n adalah jumlah pertandingan yang akan dijadwalkan. Faktor 2^n berasal dari jumlah himpunan bagian yang harus dieksplorasi, sedangkan faktor n^2 berasal dari proses validasi konflik antar pertandingan dalam setiap himpunan bagian. Algoritma ini menjamin ditemukannya solusi optimal karena seluruh kemungkinan dieksplorasi secara lengkap.

3.6. Skenario Eksperimen

Eksperimen dilakukan dengan membagi dataset ke dalam enam fase berdasarkan jumlah

pertandingan, yaitu 15, 20, 25, 30, 35, dan 70 pertandingan. Pada setiap fase, data pertandingan diacak terlebih dahulu untuk menghasilkan variasi urutan yang berbeda.
Setiap fase diuji sebanyak sepuluh kali percobaan (trial) untuk setiap algoritma. Parameter yang diukur dalam setiap percobaan meliputi waktu eksekusi dalam satuan detik, jumlah pertandingan valid yang berhasil dijadwalkan, dan jumlah konflik yang terjadi dalam jadwal yang dihasilkan.

Seluruh eksperimen dijalankan pada lingkungan cloud computing Kaggle Notebook. Spesifikasi lingkungan komputasi yang digunakan meliputi prosesor Intel Xeon dengan akselerator CPU 2,20 GHz dan RAM sebesar 16 GB. Bahasa pemrograman yang digunakan adalah Python dengan pustaka pendukung seperti pandas untuk pengolahan data, numpy untuk komputasi numerik, time untuk pengukuran waktu eksekusi, serta matplotlib untuk visualisasi hasil. Evaluasi kualitas perangkat lunak dalam penelitian ini mengacu pada standar yang telah diterapkan dalam penelitian terdahulu [15].

4. HASIL DAN PEMBAHASAN

4.1. Hasil Eksperimen Algoritma Greedy

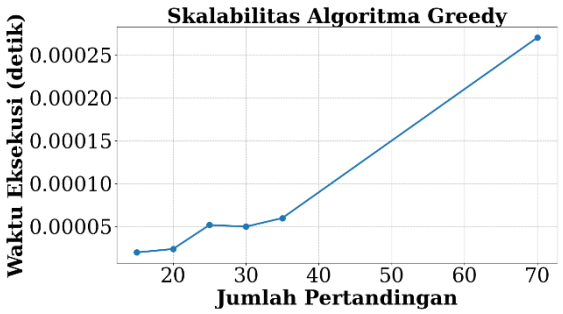
Algoritma *greedy* berhasil menyelesaikan seluruh fase pengujian dari 15 pertandingan hingga 70 pertandingan. Waktu eksekusi yang diperlukan sangat singkat dan cenderung stabil meskipun jumlah pertandingan meningkat secara signifikan. Ringkasan hasil eksperimen algoritma *greedy* disajikan pada Tabel 2.

Tabel 2. Ringkasan Hasil Eksperimen Algoritma *Greedy*

Pert.	Waktu Eksekusi (s)			Valid	Konf.
	Avg	Min	Max		
15	0,00002	0,00001	0,00003	15	0
20	0,00016	0,00002	0,00138	20	0
25	0,00005	0,00003	0,00009	25	0
30	0,00005	0,00005	0,00006	30	0
35	0,00006	0,00006	0,00010	35	0
70	0,00027	0,00021	0,00050	70	0

Berdasarkan Tabel 2, algoritma *greedy* mampu menghasilkan jadwal valid dengan jumlah pertandingan sama dengan jumlah masukan pada seluruh fase pengujian. Tingkat keberhasilan ini mencapai seratus persen. Tidak terdapat satu pun konflik yang terjadi pada jadwal yang dihasilkan, yang menunjukkan bahwa algoritma *greedy* mampu memenuhi seluruh batasan penjadwalan dengan baik.

Waktu eksekusi algoritma *greedy* pada fase 70 pertandingan tercatat sebesar 0,00027 detik. Angka ini hanya meningkat sekitar lima belas kali lipat dibandingkan fase 15 pertandingan yang memerlukan 0,00002 detik, sementara jumlah pertandingan meningkat hampir lima kali lipat. Hal ini menunjukkan bahwa algoritma *greedy* memiliki tingkat pertumbuhan waktu yang relatif landai dan bersifat polinomial, sesuai dengan persamaan (1).



Gambar 1. Skalabilitas Algoritma *Greedy*

Gambar 1 menunjukkan bahwa waktu eksekusi algoritma *greedy* cenderung stabil meskipun jumlah pertandingan meningkat dari 15 menjadi 70. Waktu eksekusi hanya meningkat dari 0,00002 detik menjadi 0,00027 detik, atau sekitar 15 kali lipat, sementara jumlah pertandingan meningkat 4,6 kali lipat. Kurva yang landai ini membuktikan bahwa algoritma *greedy* memiliki skalabilitas yang sangat baik dan tidak mengalami degradasi performa pada dataset berukuran besar.

4.2. Hasil Eksperimen Algoritma Backtracking

Algoritma *backtracking* hanya mampu menyelesaikan pengujian hingga fase 30 pertandingan. Pada fase 35 pertandingan dan 70 pertandingan, algoritma tidak dapat menyelesaikan eksekusi karena keterbatasan sumber daya komputasi. Ringkasan hasil eksperimen algoritma *backtracking* disajikan pada Tabel 3.

Tabel 3. Ringkasan Hasil Eksperimen Algoritma *Backtracking*

Pert.	Waktu Eksekusi (s)			Valid	Konf.
	Avg	Min	Max		
15	0,03	0,03	0,04	15	0
20	1,26	1,19	1,40	20	0
25	44,55	43,74	46,05	25	0

30	1.644	1.629	1.669	30	0
35	N/A[1]	N/A	N/A	N/A	N/A
70	N/A[1]	N/A	N/A	N/A	N/A

[1]Tidak dapat dieksekusi karena keterbatasan sumber daya komputasi.

Waktu eksekusi algoritma *backtracking* meningkat secara drastis seiring bertambahnya jumlah pertandingan. Pada fase 15 pertandingan, waktu yang diperlukan masih di bawah satu detik yaitu 0,03 detik. Pada fase 20 pertandingan, waktu meningkat menjadi 1,26 detik. Peningkatan paling signifikan terjadi pada fase 25 pertandingan dengan waktu mencapai 44,55 detik dan fase 30 pertandingan dengan waktu mencapai 1.644 detik.

Pola peningkatan waktu ini sesuai dengan kompleksitas teoritis algoritma *backtracking* yang bersifat eksponensial $O(2^n)$ seperti dinyatakan pada persamaan (2). Setiap penambahan jumlah pertandingan menyebabkan ruang pencarian solusi menjadi dua kali lipat lebih besar, yang berdampak langsung pada waktu yang diperlukan untuk mengeksplorasi seluruh kemungkinan.

Gambar 2. Pertumbuhan Waktu Eksekusi Algoritma *Backtracking*

Gambar 2 menunjukkan peningkatan waktu eksekusi algoritma *backtracking* secara eksponensial. Waktu eksekusi meningkat dari 0,03 detik pada 15 pertandingan menjadi 1.644 detik pada 30 pertandingan. Pola kurva yang curam ini sesuai dengan kompleksitas teoritis $O(2^n)$ pada persamaan (2).

4.3. Hasil Eksperimen Algoritma Brute Force

Algoritma *brute force* diuji sebagai pembanding untuk menunjukkan batas bawah efisiensi yang dapat dicapai. Algoritma ini hanya mampu menyelesaikan pengujian hingga fase 25 pertandingan dengan sepuluh kali percobaan. Pada fase 30 pertandingan, hanya satu kali percobaan yang berhasil diselesaikan karena waktu eksekusi yang sangat lama. Ringkasan hasil eksperimen algoritma *brute force* disajikan pada Tabel 4.

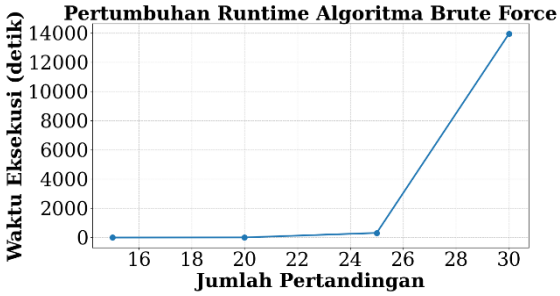
Tabel 4. Ringkasan Hasil Eksperimen Algoritma *Brute Force*

Pert.	Waktu Eksekusi (s)			Valid	Konf.
	Avg	Min	Max		
15	0,12	0,11	0,14	15	0
20	6,69	5,96	7,47	20	0
25	317	313	322	25	0
30	13.948	N/A[1]	N/A[1]	30	0

[1]Hanya satu kali percobaan yang berhasil diselesaikan.

Waktu eksekusi algoritma *brute force* meningkat sangat tajam. Pada fase 20 pertandingan, waktu yang diperlukan mencapai 6,69 detik. Pada fase 25 pertandingan, waktu melonjak menjadi 317 detik atau sekitar 5,28 menit. Pada fase 30 pertandingan, waktu yang diperlukan mencapai 13.948 detik atau sekitar 3,87 jam untuk satu kali percobaan.

Peningkatan waktu eksekusi *brute force* mengikuti kompleksitas kombinatorial di mana algoritma harus mengeksplorasi seluruh himpunan bagian dari kumpulan pertandingan, sesuai dengan persamaan (3). Jumlah himpunan bagian yang harus diperiksa pada fase 30 pertandingan adalah 2^{30} atau sekitar 1,07 miliar kombinasi.



Gambar 3. Pertumbuhan Waktu Eksekusi Algoritma *Brute Force*

Gambar 3 menunjukkan peningkatan waktu eksekusi algoritma *brute force* yang sangat tajam dan drastis. Pola kurva yang sangat curam ini sesuai dengan kompleksitas kombinatorial $O(2^n \times n^2)$ pada persamaan (3).

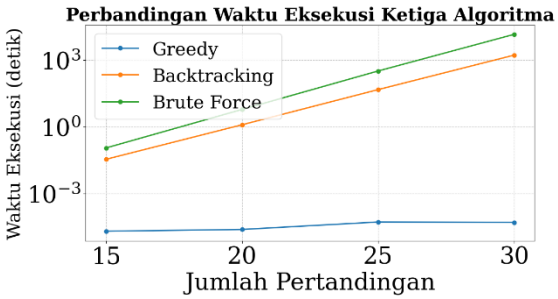
4.4. Perbandingan Performa Ketiga Algoritma

Perbandingan performa ketiga algoritma disajikan pada Tabel 5.

Tabel 5. Perbandingan Rata-Rata Waktu Eksekusi Ketiga Algoritma (detik)

Jumlah Pertandingan	Greedy	Backtracking	Brute Force
15	0,00002	0,03	0,12
20	0,00016	1,26	6,69
25	0,00005	44,55	317
30	0,00005	1.644	13.948
35	0,00006	N/A	N/A
70	0,00027	N/A	N/A

Berdasarkan Tabel 5, algoritma *greedy* secara konsisten menunjukkan waktu eksekusi tercepat pada seluruh fase pengujian. Selisih waktu antara algoritma *greedy* dan kedua algoritma lainnya semakin membesar seiring bertambahnya jumlah pertandingan. Pada fase 30 pertandingan, algoritma *greedy* menyelesaikan penjadwalan dalam waktu kurang dari satu milidetik, sementara *backtracking* memerlukan waktu 1.644 detik dan *brute force* memerlukan waktu 13.948 detik.



Gambar 4. Perbandingan Waktu Eksekusi Ketiga Algoritma

Gambar 4 menyajikan perbandingan ketiga algoritma. Kurva *greedy* berada pada posisi paling bawah dengan waktu eksekusi kurang dari 0,0003 detik dan cenderung mendatar. Kurva *backtracking* berada di posisi tengah dengan waktu eksekusi meningkat secara eksponensial. Kurva *brute force* berada pada posisi paling atas dengan waktu eksekusi meningkat sangat drastis. Lebar kesenjangan antar kurva semakin melebar seiring bertambahnya jumlah pertandingan.

4.5. Analisis Kualitas Solusi

Seluruh algoritma yang diuji dalam penelitian ini berhasil menghasilkan jadwal dengan tingkat konflik nol. Hal ini menunjukkan bahwa setiap algoritma mampu memenuhi kedua batasan penjadwalan yang telah ditetapkan, yaitu tidak terdapat bentrokan tempat pertandingan pada waktu yang sama dan tidak terdapat tim yang bertanding dua kali pada waktu yang sama.

4.6. Analisis Skalabilitas

Skalabilitas algoritma menjadi aspek penting dalam pemilihan algoritma untuk sistem penjadwalan skala besar. Berdasarkan hasil eksperimen, algoritma *greedy* menunjukkan skalabilitas terbaik karena mampu menangani peningkatan jumlah pertandingan dari 15 menjadi 70 dengan peningkatan waktu yang relatif kecil.

Sebaliknya, algoritma *backtracking* dan *brute force* menunjukkan keterbatasan skalabilitas yang serius. Peningkatan jumlah pertandingan dari 15 menjadi 30 menyebabkan peningkatan waktu eksekusi *backtracking* sebesar lebih dari 50.000 kali lipat. Untuk *brute force*, peningkatan yang sama menyebabkan peningkatan waktu eksekusi lebih dari 100.000 kali lipat.

5. KESIMPULAN

Berdasarkan hasil eksperimen dan pembahasan yang telah dilakukan, dapat ditarik beberapa kesimpulan sebagai berikut:

a. Algoritma *greedy* menunjukkan performa terbaik dari segi waktu eksekusi dengan rata-rata 0,00002 detik pada seluruh fase pengujian hingga 70 pertandingan. Algoritma ini juga memiliki skalabilitas terbaik karena mampu menangani peningkatan jumlah pertandingan tanpa peningkatan waktu yang signifikan.
b. Algoritma *backtracking* hanya mampu menyelesaikan pengujian hingga fase 30 pertandingan dengan waktu eksekusi mencapai 1.644 detik. Algoritma ini mengalami ledakan waktu eksekusi pada fase 35 dan 70 pertandingan karena kompleksitas eksponensial yang dimilikinya, sesuai dengan persamaan (2).

c. Algoritma *brute force* memiliki performa paling lambat dengan waktu eksekusi mencapai 13.948 detik pada fase 30 pertandingan. Algoritma ini terbukti tidak layak digunakan untuk dataset berukuran sedang maupun besar, sesuai dengan persamaan (3).
d. Seluruh algoritma yang diuji berhasil menghasilkan jadwal dengan tingkat konflik nol, yang menunjukkan bahwa setiap algoritma mampu memenuhi batasan penjadwalan yang telah ditetapkan.

Penelitian ini memiliki beberapa keterbatasan. Eksperimen untuk algoritma *backtracking* dan *brute force* tidak dapat diselesaikan pada fase 35 dan 70 pertandingan karena keterbatasan sumber daya komputasi.

Dataset yang digunakan hanya terbatas pada satu musim IPL yaitu tahun 2022. Penelitian ini hanya menerapkan dua batasan penjadwalan utama. Implementasi algoritma masih menggunakan pendekatan dasar tanpa optimasi tambahan.

Berdasarkan kesimpulan dan keterbatasan tersebut, algoritma *greedy* direkomendasikan untuk diterapkan pada sistem penjadwalan pertandingan IPL skala besar karena efisiensi waktu dan skalabilitas yang sangat baik. Algoritma *backtracking* dan *brute force* lebih cocok digunakan untuk keperluan akademik atau pada dataset berukuran kecil.

Untuk penelitian selanjutnya, disarankan untuk menguji ketiga algoritma pada dataset dari musim IPL yang berbeda, menambahkan lebih banyak batasan penjadwalan yang merepresentasikan kondisi nyata, serta mengoptimalkan implementasi algoritma *backtracking* dengan teknik pruning yang lebih agresif.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada pihak-pihak terkait yang telah memberi dukungan terhadap penelitian ini.

DAFTAR PUSTAKA

[1] M. Esteve, J. J. Rodriguez-Sala, J. Juan Lopez-Espin, and J. Aparicio, “Heuristic and Backtracking Algorithms for Improving the Performance of Efficiency Analysis Trees,” *IEEE Access*, vol. 9, pp. 17421–17428, 2021, doi: 10.1109/ACCESS.2021.3054006.

[2] Angel Caroline Billan and Tata Sutabri, “Restorasi Penjadwalan Sumur Minyak Yang Mengalami Off-Time Menggunakan Algoritma Backtracking Dalam Upaya Optimasi Produksi,” *Bull. Comput. Sci. Res.*, vol. 5, no. 3, pp. 228–234, Apr. 2025, doi: 10.47065/bulletincsr.v5i3.507.

[3] S. Petrova and K. Watanabe, “User-Centered Mobile Navigation: Evaluating Local Usability for Improved UX,” *J. Technol. Inform. Eng.*, vol. 4, no. 3, pp. 478–492, Dec. 2025, doi: 10.51903/jtie.v4i3.457.

[4] F. Nova Arviantino, W. Gata, L. Kurniawati, Y. A. Setiawan, and D. Priansyah, “Penerapan Algoritma Greedy Dalam Pencarian Jalur Terpendek Pada Masjid–Masjid Di Kota Samarinda,” *METIK J.*, vol. 5, no. 1, pp. 8–11, Jun. 2021, doi: 10.47002/metik.v5i1.188.

[5] A. Cantona, F. Fauziah, and W. Winarsih, “Implementasi Algoritma Dijkstra Pada Pencarian Rute Terpendek ke Museum di Jakarta,” *J. Teknol. Dan Manaj. Inform.*, vol. 6, no. 1, pp. 27–34, Apr. 2020, doi: 10.26905/jtmi.v6i1.3837.

[6] Y. Darnita and R. Toyib, “Penerapan Algoritma Greedy Dalam Pencarian Jalur Terpendek Pada Instansi-Instasi Penting Di Kota Argamakmur Kabupaten Bengkulu Utara,” *J. MEDIA INFOTAMA*, vol. 15, no. 2, Oct. 2019, doi: 10.37676/jmi.v15i2.867.

[7] N. Nurwan, W. E. Pranata, M. R. F. Payu, and N. I. Yahya, “Implementation of Dijkstra Algorithm and Welch-Powell Algorithm for Optimal Solution of Campus Bus Transportation,” *J. Mat. MANTIK*, vol. 7, no. 1, pp. 31–40, May 2021, doi: 10.15642/mantik.2021.7.1.31-40.

[8] J. S. Iskandar and Y. F. Riti, “Perbandingan Algoritma Greedy dan Algoritma Dijkstra dalam Pencarian Rute Terpendek dari Kabupaten Tuban ke Kota Surabaya”.

[9] M. Z. Usman and T. Oktiarso, “Implementasi Algoritma Greedy Untuk Menyelesaikan Travelling Salesman Problem di Distributor PT. Z,” *J. Integr. Syst.*, vol. 1, no. 2, pp. 216–229, Mar. 2019, doi: 10.28932/jis.v1i2.1049.

[10] A. A. Prsaha, C. O. Rachmadi, A. P. Sari, N. G. Raditya, and S. L. Mutiara, “Implementasi Algoritma Greedy dan Dynamic Programming untuk Masalah Penjadwalan Interval dengan Model Knapsack”.

[11] C. He, Y. Lin, and J. Yuan, "A note on the single machine scheduling to minimize the number of tardy jobs with deadlines," *Eur. J. Oper. Res.*, vol. 201, no. 3, pp. 966–970, Mar. 2010, doi: 10.1016/j.ejor.2009.05.013.

[12] F. Zhao, Y. Du, C. Zhuang, L. Wang, and Y. Yu, "An Iterative Greedy Algorithm for Solving a Multiobjective Distributed Assembly Flexible Job Shop Scheduling Problem With Fuzzy Processing Time," *IEEE Trans. Cybern.*, vol. 55, no. 5, pp. 2302–2315, May 2025, doi: 10.1109/TCYB.2025.3538007.

[13] S. S. Skiena, *The Algorithm Design Manual*. in Texts in Computer Science. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-54256-6.

[14] IPLT20, "IPL Schedule 2022." Accessed: Jun. 17, 2026. [Online]. Available: https://www.iplt20.com

[15] D. Novianti, "REDESIGN USER INTERFACE WEBSITE UNIVERSITAS BINA SARANA INFORMATIKA MENGGUNAKAN METODE DESIGN THINKING DAN SYSTEM USABILITY SCALE (SUS)," *J. Inform. Dan Tek. Elektro Terap.*, vol. 12, no. 3, Aug. 2024, doi: 10.23960/jitet.v12i3.4300.