Penyelesaian Squad Building Challenge pada Permainan FC25 dengan Penerapan Algoritma Backtracking

Kefas Kurnia Jonathan - 13523113
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: kefaskj@gmail.com , 13523113@std.stei.itb.ac.id

Abstrak*—Permainan FC25 memiliki fitur Squad Building Challenge (SBC) yang memberikan tantangan kepada pemain untuk membentuk tim berdasarkan berbagai syarat seperti rating minimum, asal negara, liga, atau klub. Permasalahan ini secara alami merupakan persoalan kombinatorial dengan banyak constraint, sehingga tidak efisien untuk diselesaikan dengan pendekatan bruteforce. Penelitian ini bertujuan untuk menerapkan algoritma runut-balik (backtracking) untuk menyelesaikan tantangan SBC secara sistematis dan efisien. Dataset yang digunakan mencakup lebih dari 18.000 pemain dari base card FC25 dengan atribut posisi, OVR, negara, liga, dan klub. Eksperimen dilakukan dengan dua skenario constraint yang berbeda, dan hasil menunjukkan bahwa algoritma backtracking mampu menemukan solusi yang valid dalam waktu yang singkat (kurang dari 0.1 detik), meskipun jumlah simpul yang dieksplorasi mencapai ratusan ribu. Eksperimen juga menunjukkan pengaruh jenis constraint terhadap kompleksitas pencarian. Penelitian ini membuktikan bahwa backtracking efektif digunakan dalam konteks pemilihan tim dengan syarat majemuk dan dapat diperluas untuk simulasi yang lebih kompleks di masa depan.

Kata Kunci*—algoritma backtracking, squad building challenge, FC25, constraint satisfication, pemilihan kombinatorial.

I. PENDAHULUAN

Permainan video sepak bola saat ini sudah berkembang sangat pesat. Dari sebuah simulasi pertandingan, kini menjadi sebuah pengalaman manajemen tim yang kompleks, membentuk sebuah tim bahkan bermain sebagai seorang ataupun tim. Salah satu fitur utama dalam permainan FC24 dan penerusnya, FC25 adalah *FIFA Ultimate Team* (FUT), yang memungkinkan pemain membangun tim impian mereka dari berbagai pemain sepak bola di seluruh dunia. Salah satu tantangan yang populer dalam mode permainan ini adalah *Squad Building Challenge* (SBC), yaitu sebuah tantangan yang meminta pemain untuk menyusun *squad* berdasarkan syarat-syarat tertentu, seperti nilai rata-rata OVR minimal, keberadaan pemain dari negara atau klub tertentu, serta batasan jumlah pemain dari liga tertentu di belahan dunia. Tampak sederhana, namun tantangan ini seringkali bersifat kombinatorial dan memerlukan pertimbangan yang kompleks untuk diselesaikan secara efisien.

Squad Building Challenge mengharuskan pemain untuk memilih kombinasi sebelas pemain dari kumpulan kartu pemain yang dimiliki, dengan memperhatikan berbagai *constraint* yang bersifat ketat. Constraint yang diminta bermacam-macam, dapat berupa kuantitatif (misalnya total OVR tim harus lebih besar dari 85), syarat atribut (misalnya minimal satu pemain dari negara Brasil), maupun syarat struktur (misalnya semua posisi harus terisi secara alami). Dengan banyaknya kemungkinan kombinasi pemain yang tersedia, penyelesaian masalah ini secara brute-force akan memakan waktu yang sangat lama. Oleh karena itu, diperlukan sebuah pendekatan algoritma yang mampu untuk mengeksplor seluruh solusi secara efisien, tanpa harus memerika kemungkinan secara menyeluruh.

Salah satu pendekatan algoritma yang dapat diterapkan untuk menyelesaikan permasalahan ini adalah algoritma runut-balik (*backtracking*) yang dipelajari pada mata kuliah IF2211 – Strategi Algoritma. *Backtracking* merupakan teknik eksplorasi solusi secara rekursif dan cocok diimplementasikan untuk menyelesaikan masalah dengan ruang solusi yang besar, namun masih dapat dipangkas ketika solusi sudah tidak mungkin memenuhi *constraint*. Dalam konteks ini, algoritma akan mencoba memasukkan pemain ke dalam skuad, hingga tidak mungkin memenuhi syarat, maka akan mundur dan mencoba alternatif lainnya. Dengan demikian, penelitian ini bertujuan untuk menerapkan algoritma *backtracking* dalam tantangan *Squad Building Challenge* pada permainan FC24/FC25 dengan memodelkan pemain dengan atribut-atribut yang relevan.

Makalah ini terdiri dari beberapa bagian, bagian utama yaitu pendahuluan, kemudian landasan teori yang mencakup pengertian algoritma *backtracking* dan penjelasan lebih dalam mengenai struktur dan aturan dalam tantangan SBC. Setelah itu, dilanjutkan ke metode penelitian dan hasil analisisnya, diikuti beberapa eksperimen sederhana menggunakan dataset pemain yang ada. Terakhir, makalah ini ditutup dengan kesimpulan dan saran untuk pengembangan yang lebih lanjut. Dengan demikian, diharapkan penelitian ini tidak hanya memberikan kontribusi akademik, melainkan memberikan wawasan menarik terkait konsep informatika yang diaplikasikan dalam dunia hiburan.

II. LANDASAN TEORI

A. Algoritma Runut Balik (*Backtracking*)

A1. Definisi dan Konsep Dasar

Algoritma runut-balik atau *backtracking* merupakan suatu teknik pencarian solusi yang sistematis dalam ruang solusi yang besar. Backtracking dapat dipandang sebagai sebuah fase dalam algoritma traversal DFS, atau sebagai sebuah metode pemecahan masalah yang sangkil, terstruktur, dan sistematis, baik untuk persoalan optimasi, maupun non-optimasi. Dengan demikian, *backtracking* merupakan tindakan lanjut dari *exhaustive search*. Ini ditunjukkan dengan tidak mengecek semua kemungkinan solusi dieksplorasi dan dievaluasi satu-satu, melainkan hanya pilihan yang mengarah ke solusi yang dieksplorasi, dengan teknik *pruning* simpul-simpul yang tidak mengarah ke solusi.

A2. Ruang Solusi dan Pohon Status

Solusi dalam algoritma *backtracking* dinyatakan sebagai vektor n-tuple: $X = (x_1, x_2, \ldots, x_n)$, dimana setiap x_i diambil dari suatu himpunan S_i. Dalam konteks *Squad Building Challenge*, x_i dapat direpresentasikan sebagai pemain ke-i dalam tim yang dibentuk. Ruang solusi ini diorganisasikan dalam bentuk pohon status (state-space tree), yang setiap simpul menyatakan solusi parsial, dan setiap lintasan dari akar ke daun menyatakan solusi lengkap. Algoritma ini secara eksplisit atau implisit menelurusi pohon ini sambil menerapkan fungsi pembatas untuk menghindari eksplorasi terhadap simpul yang tidak valid (*pruning*).

A3. Fungsi Pembangkitan dan Fungsi Pembatas

Dalam algoritma runut-balik, terdapat dua fungsi pentingm yaitu fungsi pembangkitan dan fungsi pembatas. Fungsi pembangkitan (*generating function*) $T(\ldots)$ digunakan untuk membangkitkan kandidat nilai untuk komponen solusi selanjutnya. Sedangkan fungsi pembatas (*bounding function*) $B(\ldots)$ berfungsi untuk memeriksa apakah solusi parsial saat ini masih memungkinkan untuk menghasilkan solusi akhir yang valid atau tidak. Jika $B(\ldots)$ bernilai salah, maka algoritma *backtracking* akan berhenti pada cabang tersebut, dan kembali ke langkah selanjutnya.

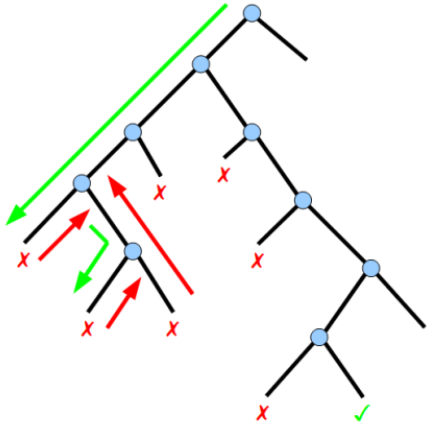
A4. Kompleksitas Waktu dan Strategi Pruning

Dalam kasus terburuk, algoritma *backtracking* tetap membutuhkan waktu yang eksponensial karena tetap dapat mengeksplorasi seluruh kemungkinan kombinasi. Untuk n elemen, kompleksitas bisa mencapai $O(2^n)$ atau $O(n!)$, tergantung seberapa kompleks masalah yang ingin diselesaikan. Oleh karena itu, strategi *pruning* menjadi aspek yang sangat penting dalam algoritma ini. Semakin awal simpul tidak valid dapat dikenali, maka semakin besar pula ruang solusi yang dapat dihindari untuk dieksplorasi, sehingga kompleksitas waktu dapat berkurang.

A5. Prinsip Pencarian Solusi dengan Algoritma Backtracking

Pencarian solusi dalam algoritma ini dapat ditelurusi sebagai berikut. Solusi dicari dengan membangkitkan simpul-simpul status, sehingga menghasilkan suatu lintasan dari akar ke daun. Aturan pembangkitan simpul yang digunakan pada algoritma ini mengikuti aturan DFS (*depth-first-order*), dan simpul-simpul ini dinamakan simpul hidup (*live node*). Simpul hidup yang diperluas dinamai simpul-E (*expand-node*), dan setiap kali simpul-E diperluas, maka lintasan yang dibangun olehnya akan bertambah panjang.

Ketika suatu simpul diperluas, jika lintasan yang terbentuk tidak mengarah ke solusi, maka simpul tersebut "dimatikan", sehingga menjadi simpul mati (*dead node*). Fungsi yang digunakan untuk mengecek apakah simpul ini patut dimatikan atau tidak adalah fungsi pembatas yang dibahas pada sub-poin sebelumnya. Ketika sebuah simpul dinyatakan mati, maka kita telah memangkas (*pruning*) simpul anak-anaknya. Jika pembentukan lintasan berakhir dengan simpul mati, maka akan *backtrack* ke simpul atasnya. Hal ini diteruskan terus dengan membangkitkan simpul anak lainnya, dan pencarian dihentikan bila telah sampai pada *goal node* atau tujuan.



Gambar 1. Visualisasi Pencarian Solusi dengan *Backtrackings*
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/15-Algoritma-backtracking-(2025)-Bagian1.pdf diakses pada 22/06/2025

B. Squad Building Challenge (SBC) dalam permainan FC25

B1. Deskripsi Umum SBC

Squad Building Challenge (SBC) merupakan salah satu fitur utama dalam mode permainan *FIFA Ultimate Team* (FUT) pada permainan *EA Sports FC25* dan tahun-tahun sebelumnya. Dalam fitur ini, pemain diminta untuk menyusun tim sepak bola yang terdiri dari 11 pemain, dengan syarat-syarat atau *constraint* khusus yang ditentukan oleh sistem, tergantung dari setiap tantangan yang disediakan. Syarat-syarat ini dapat berupa:

- Minimum *Overall Rating* (OVR) yang dibentuk oleh tim tersebut
- Minimum jumlah pemain dari suatu negara tertentu seperti negara Brazil atau lainnya.
- Minimum jumlah pemain dari suatu klub tertentu seperti Real Madrid dan lainnya
- Minimum chemistry atau kesesuaian posisi pemain dalam tim tersebut
- Minimum pemain dari *event* khusus, seperti *Team of The Week*, *Team of The Season*, atau *Icon*.

Gambar 2. Contoh Tantangan SBC pada Permainan FC25
Sumber: UniqueRiggers on Youtube, https://www.youtube.com/watch?v=lDJdr7kvKvo, diakses pada 22/06/25

Fitur *Squad Building Challenge* dirancang untuk mendorong pemain menggunakan kreativitas dan strategi untuk memilih pemain yang sesuai hingga akhirnya bisa menyelesaikan tantangan yang diberikan. Ini juga menjadi sebuah sarana untuk mengurangi stok pemain berlebih yang tidak terpakai dalam klub yang dimiliki. Biasanya SBC terbatas oleh waktu dan disertai hadiah yang sangat menarik, seperti paket pemain, kartu spesial, atau pemain eksklusif yang hanya bisa didapatkan dari mode ini. Dengan demikian, menyelesaikan SBC secara efisien menjadi salah satu aspek kompetitif yang penting dalam pengalaman bermain FUT.

B2. Karakteristik Masalah SBC dari Perspektif Informatika

Melihat dari sudut pandang informatika, terlebih dalam menentukan suatu strategi algoritma, SBC merupakan contoh nyata dari sebuah masalah kombinatorial dengan banyak *constraint*, yang bisa dimodelkan sebagai *Constraint Satisfication Problem* (CSP). Pemain harus mencari satu (atau beberapa) kombinasi dari sekian banyak kemungkinan yang memenuhi semua syarat tersebut secara bersamaan.

Dalam konteks algoritma *backtracking*, tantangan SBC dapat diformulasikan sebagai:

- Ruang solusi *s* adalah himpunan semua kombinasi 11 pemain dari koleksi pemain bola yang pemain miliki
- Fungsi objektif untuk menyusun kombinasi valid yang memenuhi *constraint*, tanpa perlu mengoptimasi nilai tertentu (non-optimasi)
- *Constraint* dapat berbentuk *hard constraint* (harus dipenuhi) seperti, "harus ada minimal 2 pemain dari La Liga", atau *soft constraint* seperti "OVR ≥ 85".

Ukuran ruang solusi bertambah secara secara eksponensial terhadap banyaknya pemain yang dimiliki. Hal ini membuat pendekatan dengan algoritma *bruteforce* menjadi tidak efisien dalam skala besar, sehingga diperlukan sebuah strategi yang cerdas dan efisien, yaitu *backtracking*.

C. Penerapan Algoritma *Backtracking pada* SBC

C1. Representasi Solusi

Penerapan algoritma *backtracking* pada *Squad Building Challenge* (SBC) dapat dimulai dengan merepresentasikan solusi sebagai sebuah vektor $X = (x_1, x_2, \ldots, x_n)$, dimana setiap x_i merepresentasikan seorang pemain yang akan ditempatkan pada posisi tertentu dalam formasi tim, sesuai kebutuhan tantangan. Setiap pemain memiliki atribut-atribut seperti posisi, klub, negara, *overall rating* (OVR), dan kategori kartu. Pemilihan pemain dilakukan satu per satu, dan pada setiap tahap (*depth* dalam rekursi), sistem akan memutuskan apakah pemain yang saat ini dipilih membuat solusi parsial tetap valid atau tidak. Sebagai contoh:

- x_1 = pemain pertama, harus dari negara Jerman
- x_2 = pemain kedua, harus dari Ligue 1 Perancis, dan seterusnya

Jika pada suatu tahap pemilihan pemain tidak memenuhi *constraint* (misalnya rating terlalu rendah atau posisi sudah terisi), maka pencarian akan dihentikan pada titik itu, dan *backtrack* (kembali) ke tahap sebelumnya.

C2. Fungsi Pembangkitan

Fungsi pembangkitan $T(\ldots)$ berperan untuk membangkitkan kandidat pemain berikutnya dari kumpulan pemain yang tersedia. Biasanya, pemain akan dipilih dari kumpulan yang belum pernah digunakan untuk memastikan tidak ada duplikasi, memiliki atribut potensial untuk memenuhi constraint (misalnya OVR tinggi, atau berasal dari negara/liga/klub yang dibutuhkan), bisa juga sesuai dengan posisi yang dibutuhkan jika terdapat *constraint* posisi harus sama dengan formasi.

C3. Fungsi Pembatas

Fungsi pembatas $B(\ldots)$ berfungsi untuk mengevaluasi apakah solusi parsial sementara $x_1, x_2, \ldots, x_k$ masih mungkin untuk menjadi solusi akhir yang valid. Fungsi ini adalah salah satu fungsi krusial untuk memangkas ruang solusi agar tidak perlu memeriksa seluruh kemungkinan. Beberapa contoh fungsi pembatas pada persoalan SBC ini meliputi:

1. Apakah total rata-rata OVR masih bisa mencapai batas minimum?
2. Apakah masih mungkin memenuhi jumlah minimum pemain dari negara/liga/klub tertentu dengan sisa slot yang tersedia?
3. Apakah posisi yang sedang diisi belum overload?

Sehingga, jika fungsi pembatas ini mengembalikan nilai yang bernilai "salah", maka simpul tersebut dan seluruh turunannya tidak perlu ditelusuri (*pruning*).

C4. Proses Rekursi dan Backtrack

Proses rekursi untuk pemecahan masalah ini dimulai dari posisi pertama dan berjalan hingga posisi ke-*n* sesuai jumlah pemain yang diminta oleh tantangan. Pada setiap level, pemain yang sesuai dicoba untuk dimasukkan ke dalam *squad*. Jika setelah memasukkan pemain tertentu solusi menjadi tidak valid (misalnya OVR terlalu rendah atau *constraint* lainnya dilanggar), maka sistem akan menghapus pemian tersebut dari *squad* sementara, mencoba kandidat lain pada posisi yang sama, dan jika tidak ada kandidat yang tersisa, akan kembali (*backtrack*) ke posisi sebelumnya. Hal ini dapat divisualisasikan sebagai pohon status yang setiap simpulnya

merepresentasikan sebuah pilihan pemain, dan cabangnya adalah kelanjutan kombinasi yang tersedia.

III. Metode Penelitian

A. *Data dan Variabel*

Penelitian ini menggunakan dataset seluruh pemain yang tersedia pada permainan FC25 dalam kondisi *base* atau kartu dasar. Data yang dimiliki mencakup:

- Nama pemain
- *Overall rating* pemain
- Posisi pemain seperti ST, LW, CM, CB, GK, dll.
- Kebangsaan untuk *constraint* negara
- Liga tempat pemain bermain
- Klub yang dibela pemain

Dengan data yang dimiliki ini, dapat digunakan sebagai ruang solusi maksimal yang dapat digunakan untuk menguji implementasi algoritma *backtracking* dalam menyelesaikan masalah.

Dataset ini diambil dari pranala GitHub milik *prashantghimire*, yang dapat diakses pada https://github.com/prashantghimire/sofifa-web-scraper. Dataset ini menyediakan lebih dari 18.000 informasi pemain dari EA FC25. Untuk keperluan pengujian algoritma *backtracking*, maka hanya 6 atribut yang digunakan. Beberapa contoh isi dataset untuk pemain bola pada permainan FC25 dapat dilihat sebagai berikut.

Tabel 1. Contoh Dataset yang Digunakan

No	Nama	OVR	Posisi	Nation	Liga	Klub
1	Erling Haaland	91	ST	Norway	Premier League	Real Madrid
2	Kylian Mbappe	91	ST	France	Ligue 1	PSG
3	Kevin De Bruyne	91	CM	Belgium	Premier League	Manchester City
4	Rodri	90	CDM	Spain	Premier League	Manchester City
5	Harry Kane	90	ST	England	Premier League	Bayern Munchen
6	Thibaut Courtois	90	GK	Belgium	La Liga	Real Madrid
7	Robert Lewandowski	90	ST	Poland	La Liga	FC Barcelona
8	Lionel Messi	90	CF	Argentina	MLS	Inter Miami
9	Ruben Dias	89	CB	Portugal	Premier League	Manchester City
10	Vini Jr.	89	LW	Brasil	La Liga	Real Madrid

B. *Desain Eksperimen*

Eksperimen dilakukan menggunakan program bantuan yang dibuat menggunakan bahasa pemrograman Python. Program ini akan membaca dataset dalam bentuk CSV, menggunakan modul *csv* untuk membaca file CSV, dan *time* untuk menghitung waktu eksekusi. Program ini terbagi menjadi dua kelas utama, yaitu kelas Player yang berisikan data suatu pemian, meliputi nama, OVR, posisi yang bisa dimainkan, klub, negara, dan juga liga tempat dia bermain. Serta kelas SquadBuildingChallenge yang digunakan untuk menyelesaikan masalah utama ini.

Pada eksperimen, karena keterbatasan, hanya menggunakan contoh tantangan dengan jumlah pemain pada tim lengkap (11 pemain), dengan formasi normal, yaitu formasi 4-3-3, dengan posisi sebagai berikut: ['GK', 'LB', 'CB', 'CB', 'RB', 'CM', 'CM', 'CM', 'LW', 'ST', 'RW']. Program dapat menerima masukan dari user untuk menentukan apakah pemain harus sesuai dengan posisi mereka atau tidak. Selain itu, program dapat menerima masukan berupa pilihan *constraint* yang dapat berlaku untuk suatu tantangan, seperti minimal jumlah pemain dari suatu negara, atau minimal OVR, atau minimal jumlah pemain dari klub tertentu, ataupun minimal jumlah pemain dari liga tertentu. Kemudian, program akan memberikan hasil yang memenuhi seluruh *constraint* tersebut menggunakan algoritma *backtracking*, beserta jumlah *nodes visited*, serta waktu pencarian.

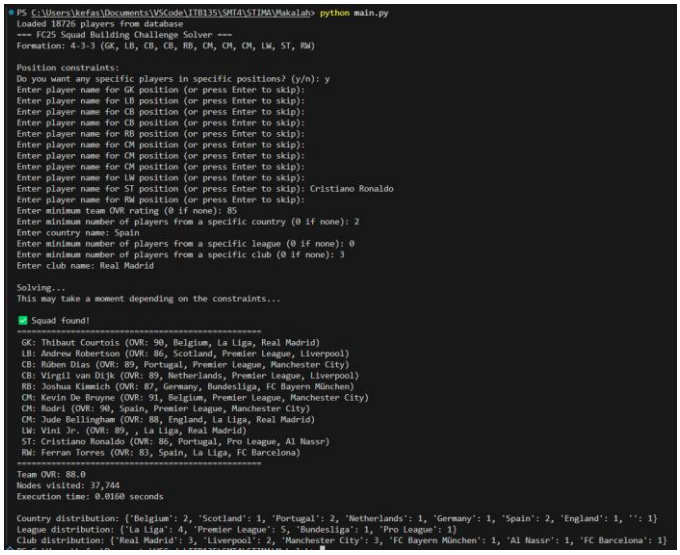
C. *Contoh Eksperimen*

Eksperimen yang pertama kali dicoba adalah mencari sebuah formasi tim yang mengharuskan tim yang dibentuk memiliki OVR (rata-rata rating) minimal 85, dengan minimal 3 pemain berasal dari negara Spanyol, dan minimal 1 pemain berasal dari Ligue 1 Perancis, dan minimal 2 pemain berasal dari klub Real Madrid.

```
PS C:\Users\kefas\Documents\VSCode\ITB135\SMT4\STIMA\Makalah> python main.py
Loaded 18726 players from database
=== FC25 Squad Building Challenge Solver ===
Formation: 4-3-3 (GK, LB, CB, CB, RB, CM, CM, CM, LW, ST, RW)

Position constraints:
Do you want any specific players in specific positions? (y/n): n
Enter minimum team OVR rating (0 if none): 85
Enter minimum number of players from a specific country (0 if none): 3
Enter country name: Spain
Enter minimum number of players from a specific league (0 if none): 1
Enter league name: Ligue 1
Enter minimum number of players from a specific club (0 if none): 2
Enter club name: Real Madrid

Solving...
This may take a moment depending on the constraints...

✅ Squad found!
============================================
 GK: Thibaut Courtois (OVR: 90, Belgium, La Liga, Real Madrid)
 LB: Andrew Robertson (OVR: 86, Scotland, Premier League, Liverpool)
 CB: Rúben Dias (OVR: 89, Portugal, Premier League, Manchester City)
 CB: Virgil van Dijk (OVR: 89, Netherlands, Premier League, Liverpool)
 RB: Joshua Kimmich (OVR: 87, Germany, Bundesliga, FC Bayern München)
 CM: Kevin De Bruyne (OVR: 91, Belgium, Premier League, Manchester City)
 CM: Rodri (OVR: 90, Spain, Premier League, Manchester City)
 CM: Jude Bellingham (OVR: 88, England, La Liga, Real Madrid)
 LW: Kylian Mbappé (OVR: 91, France, Ligue 1, Paris Saint Germain)
 ST: Morata (OVR: 84, Spain, La Liga, Atlético Madrid)
 RW: Ferran Torres (OVR: 83, Spain, La Liga, FC Barcelona)
============================================
Team OVR: 88.0
Nodes visited: 299,805
Execution time: 0.0942 seconds

Country distribution: {'Belgium': 2, 'Scotland': 1, 'Portugal': 1, 'Netherlands': 1, 'Germany': 1, 'Spain':
3, 'England': 1, 'France': 1}
League distribution: {'La Liga': 4, 'Premier League': 5, 'Bundesliga': 1, 'Ligue 1': 1}
Club distribution: {'Real Madrid': 2, 'Liverpool': 2, 'Manchester City': 3, 'FC Bayern München': 1, 'Paris S
aint Germain': 1, 'Atlético Madrid': 1, 'FC Barcelona': 1}
```

Gambar 3. Contoh Tantangan SBC pada Permainan FC25

Eksperimen selanjutnya dapat dilihat pada gambar berikut, dengan mencoba salah satu *constraint* yang berbeda:



Gambar 3. Contoh Tantangan SBC pada Permainan FC25

Constraint yang digunakan pada eksperimen kali ini adalah wajib menggunakan pemain dengan nama "Cristiano Ronaldo" pada posisi ST (striker), disertai dengan minimal OVR tim 85, dan minimal 2 pemain berasal dari negara Spanyol, serta 3 pemain berasal minimal dari klub Real Madrid.

IV. Hasil dan Pembahasan

A. Hasil Eksperimen

Berdasarkan kedua eksperimen yang dapat dilihat pada bagian Metode Penelitian, eksperimen dilakukan dengan dua skenario tantangan SBC yang berbeda, dengan masing-masing memiliki kombinasi *constraint* yang realistis seperti yang biasa dijumpai pada permainan FC25. Eksperimen ini dijalankan pada sebuah sistem dengan formasi tetap 4-3-3, dan pemain dipilih berdasarkan dataset yang berisi lebih dari 18.000 pemain berdasarkan *base card* FC25.

Eksperimen pertama dengan *constraint* minimal OVR tim 85, minimal 3 pemain dari negara Spanyol, minimal 1 pemain dari liga Ligue 1, dan minimal 2 pemain dari klub Real Madrid, memberikan hasil yang konkrit. Hasil pencarian menggunakan *backtracking* berhasil menemukan tim yang valid dengan waktu eksekusi hanya 0.0942 detik dan mengunjungi 299.805 node. Rata-rata OVR tim yang didapatkan berada pada nilai 88.0, yang secara signifikan melebihi *constraint* yang diminta. Distribusi negara, liga, dan klub juga terpenuhi seperti penjelasan yang ada pada gambar hasil eksekusi program di bab sebelumnya.

Eksperimen kedua dilakukan sedikit berbeda dengan *constraint* spesifik posisi, yaitu posisi striker (ST) harus diisi dengan pemain bernama Cristiano Ronaldo, dengan *constraint* tambahan berupa minimal OVR tim 85, minimal 2 pemain dari Spanyol, serta minimal 3 pemain dari klub Real Madrid. Eksperimen ini menunjukkan bahwa program yang dibuat juga mampu menerima *constraint* spesifik per posisi dan tetap mampu menghasilkan tim yang valid. Algoritma *backtracking* menemukan solusi dengan waktu 0.016 detik dan 37.744 node yang dikunjungi. Ini menunjukkan bahwa keberadaan *constraint* posisi yang harus ditempati, secara langsung mempersempit ruang solusi, sehingga waktu pencarian dan node yang dikunjungi berkurang dibandingkan eksperimen pertama.

B. Pembahasan Lainnya

Berdasarkan dua eksperimen tersebut, dapat dilihat bahwa jumlah simpul yang dikunjungi (*nodes visited*) dapat bervariasi, tergantung pada jumlah dan jenis *constraint* yang diberikan. Jika *constraint* bersifat global (seperti jumlah pemain dari negara/klub), ini membuat algoritma harus mencari hingga akhir (semua posisi terisi), lalu melakukan pengecekan akhir. Akan tetapi, *constraint* yang bersifat lokal atau spesifik untuk suatu posisi, dapat mempersempit pencarian lebih awal saat pemilihan pemain untuk suatu posisi. Meskipun menggunakan dataset yang besar, jika *constraint* sesuai, rata-rata eksekusi dapat tetap rendah (kurang dari 0.1 detik), karena efisiensi fungsi pembatas pada program yang dibuat. Ini menunjukkan keunggulan *backtracking* yang mampu untuk memangkas eksplorasi cabang (*pruning*).

Meskipun demikian, pada percobaan lain yang tidak dicantumkan pada bab ketiga, terdapat percobaan seperti misalnya mencari tim dengan minimal 10 pemain dari negara inggris. Hasilnya menunjukkan bahwa algoritma memerlukan waktu yang sangat lama karena beberapa hal. Salah satunya, karena tidak semua posisi dapat diisi oleh pemain asal Inggris, atau bisa saja *constraint* tersebut terlalu sempit jika dibandingkan dengan distirbusi pemain pada *dataset*, menyebabkan eksplorasi mendekati kasus *bruteforce*. Eksperimen ini kemudian dihentikan oleh penulis sebelum solusi ditemukan, yang mengindikasikan bahwa *constraint* terlalu berat atau tidak realistis, dan diperlukan pengembangan lanjutan seperti heuristik atau *prefiltering* untuk mengecek keberadaan solusi.

V. Kesimpulan

Berdasarkan hasil eksperimen dan implementasi yang dilakukan, dapat disimpulkan bahwa algoritma backtracking terbukti mampu untuk menyelesaikan tantangan Squad Building Chalelnge (SBC) pada permainan FC25 dengan efektif. Permasalahan SBC secara struktural merupakan *constraint satisfication problem* (CSP) yang dapat dimodelkan dan dieksplorasi dengan metode runut-balik secara rekursif, yang mampu menghindari eksplorasi solusi tidak valid melalui teknik *pruning*.

Program yang dibangun berhasil memenuhi berbagai skenario constraint yang melibatkan batasan posisi, overall rating, jumlah pemain dari negara tertentu, serta jumlah pemain dari liga atau klub tertentu. Pada kasus dengan *constraint* berat namun realistis, solusi ditemukan dalam waktu sangat singkat (kurang dari 0.1 detik), meskipun simpul yang dievaluasi dapat mencapai ratusan ribu. Ini menunjukkan bahwa efisiensi pencarian sangat bergantung pada *constraint* yang diberikan.

Eksperimen tambahan juga menunjukkan bahwa *constraint* yang terlalu sempit atau berat, secara signifikan dapat memperbesar ruang solusi, bahkan menyebabkan proses pencarian membutuhkan waktu yang sangat lama atau gagal menemukan solusi tanpa optimasi tambahan

VI. Saran

Terdapat beberapa saran dan potensi untuk pengembangan selanjutnya terkait penelitian ini. Pertama, implementasi dapat diperluas dengan memasukkan pemain kartu khusus seperti ICON, *Team of the Week*, *Team of the Season*, dan lain-lain sesuai dengan perkembangan kategori kartu yang terdapat pada FIFA Ultimate Team. Melalui penambahan ini, simulasi SBC menggunakan algoritma *backtracking* lebih realistis, sesuai kondisi aktual dalam permainan FC25.

Selain itu, untuk mempercepat pencarian solusi, dalam algoritma yang dibangun dapat dilakukan penambahan heuristik, seperti seleksi pemain awal berdasarkan rating tertinggi atau asal negara yang diutamakan untuk mempercepat pencarian solusi dan mengurangi jumlah simpul yang dikunjungi. Dapat pula dilakukan *prefiltering constraint* untuk memeriksa apakah *constraint* yang dimasukkan realistis dan dapat dipenuhi dari dataset yang tersedia.

Terakhir, program dapat dikembangkan menjadi sebuah aplikasi final dengan antarmuka pengguna visual yang memungkinkan pengguna untuk melakukan simulasi SBC secara interaktif, serta penambahan beberapa algoritma pencarian lainnya untuk efisiensi lebih tinggi pada *constraint* yang kompleks, seperti algoritma *branch and bound* atau A*. Dengan demikian, sistem dapat digunakan sebagai asisten pintar bagi pemain FC25 dalam merancang strategi tim untuk menyelesaikan tantangan SBC.

VII. Ucapan Terima Kasih

Pertama-tama, saya mengucapkan puji syukur setinggi-tingginya kepada Tuhan Yang Maha Esa karena sudah memberikan saya kekuatan dan kesempatan untuk menyelesaikan makalah dengan judul "Penyelesaian Squad Building Challenge pada Permainan FC25 dengan Penerapan Algoritma Backtracking". Terima kasih juga saya ucapkan kepada keluarga yang selalu menjadi motivasi penulis dan menjadi dorongan untuk terus berjuang sampai akhir. Saya juga berterima kasih kepada Dr. Ir. Rinaldi, M.T selaku dosen mata kuliah IF2211 Strategi Algoritma atas jasanya yang telah mengajar saya dengan begitu semangat dan luar biasa. Terima kasih untuk seluruh pihak yang membantu saya dalam penulisan makalah ini. Terakhir, seperti biasa, terima kasih untuk Real Madrid meskipun tahun ini puasa *major title*, tetap selalu ada di hati, Hala Madrid!

Lampiran

Dataset *base card* seluruh pemain dalam permainan FC25 dapat dilihat pada GitHub akun milik *prashantghimire*, yang dapat diakses pada https://github.com/prashantghimire/sofifa-web-scraper.

Referensi

[1] Munir, Rinaldi. "IF2211 Strategi Algoritma – Semester 2 Tahun 2024/2025", 2025. Tersedia pada: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/stima24-25.htm, diakses pada 22/06/2025.

[2] Brailsford, Sally C., Chris N. Potts, and Barbara M. Smith. "Constraint satisfaction problems: Algorithms and applications." *European journal of operational research* 119.3 (1999): 557-581.I.S. Jacobs and C.P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350, diakses pada 23/06/2025.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 23 Juni 2025

Kefas Kurnia Jonathan - 13523113