Perbandingan Kinerja Algoritma BFS dan A* untuk Auto-Routing pada Desain Fisik Chip (Integrated Circuit)

Anas Fathurrahman - 13222033
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: anasfathurrahman.edu@gmail.com , 13222033@std.stei.itb.ac.id

***Abstract*—Auto-routing merupakan tahapan penting dalam proses desain rangkaian sirkuit terpadu (*integrated circuit*). Proses ini biasanya berada pada fase PnR (*Placement and Routing)* yang menerjemahkan koneksi nodal dari ratusan hingga jutaan koneksi *cell*. Pada makalah ini membandingkan performa dari algortima A* dan BFS dalam efisiensi jumlah iterasi dan waktu dalam menentukan koneksi antas nodal. Pada simulasi yang dilakukan, menggunakan acuan teknologi PDK SKY130 yang disederhanakan. Uji coba dilakukan pada petak 3 dimensi berukuran 360 x 360 x 9. Lapisan yang digunakan atas 5 metal dan 4 via. Hasil eksperimen menunjukan bahwa kedua algoritma memberikan hasil yang hampir setara optimalnya, namun A* hanya membuka 5.5% simpul dibanding BFS. Namun secara waktu komputasi, perhitungan heuristik pada A* menyumbang durasi yang cukup besar pada jalur 3 dimensi dan DRC yang ditetapkan. Menunjukkan bahwa efisiensi ruang tidak serta merta menunjukkan efisiensi domain waktu.**

Keywords—auto-routing, desain IC, BFS, A*, heuristik, design rule checking, SKY130, pencarian lintasan

I. PENDAHULUAN

Pada alur perancangan rangkaian terpadu modern, tahap desain *layout* rangkaian untuk diterapkan secara fisik pada *silicon wafer* dibantu oleh perangkat lunak yang melakukan sintetik serta PnR (*Placement and Routing*). Pada tahap PnR, perangkat lunak akan melakukan penyusunan komponen rangkaian secara geometris memenuhi persayaratan dari PDK (*Process Design Kit*). Set aturan ini disebut sebagai *Design Rule,* umumnya digunakan oleh perangkat lunak DRC (*Design Rule Check)* untuk memastikan bahwa peletakan dan *routing* memenuhi persyaratan fabrikasi. Ketika menerapkan PnR, algoritma komputer akan menyelesaikan pencarian susunan peletakan yang memenuhi konstrain yang diterapkan PDK. KOnstrain yang ditetapkan antara lain seperti minimum lebar koneksi, jarak minimum antar komponen, serta aturan peletakan lapisan dari *wafer* [1].

Terdapat dua bagian permasalahan yang perlu diselesaikan oleh algoritma, yakni bagaimana setiap *standard cell* disusun dan diletakkan pada area yang disediakan, dan bagaimana *netlist* saling dihubungkan. Proses ini merupakan iterasi yang sangat berat terutama pada rangkaian terpadu moderen yang memiliki jutaan *netlist*. Pada makalah ini memberikan batasan pada proses bagaimana netlist saling dihubungkan secara efisien dengan diberikan penempatan awal dari *standard cell*. Proses menghubungkan *netlist* ini disebut dengan *routing*.

Salah satu model klasik yang cukup populer untuk *routing* satu net adalah *maze routing* yang dipopulerkan oleh Lee[2]. Secara dasar ini merupakan algoritma pencarian rute terpendek pada bidang grid 2 dimensi. Pada publikasi asli yang diterbitkan oleh Lee, pencarian jalur termurah menggunakan algoritma dasar BFS (*Breadth-FIrst Search*) yang memastikan mendapatkan solusi terbaik. Namun secara konsep persoalan, algoritma lain seperti A*, DFS, ataupun Djikstra dapat digunakan.

Algoritma klasik menggunakan DFS menawarkan jalur yang dipastikan paling murah, namun pada kasus tertentu secara hipotesis awal dapat menjadikan beban komputasi yang sangat masif pada desain rangkaian yang sangat kompleks. Pada percobaan ini mengujikan perban dingan penggunaan DFS dan A***. A*** digunakan sebagai pembanding dikarenakan karakteristiknya yang menggunakan heuristik yang menawarkan pencaraian yang lebih terarah ke arah tujuan sambungan *netlist*[3]. Sehingga memunculkan rumusan masalah terkait bagaimana performa kedua algoritma dalam penyelesaian domain yang menjadi dasar perkembangan teknologi modern saat ini.

Pada pengujian yang dilakukan pada makalah ini mengambil referensi teknologi nyata fabrikasi semikonduktor yang bersifat *open-source* yakni PDK SKY130 (*Sky Water*). Teknologi ini adalah PDK sumber terbuka hasil kolaborasi Google dengan SkyWater Technology. Pada pengujian ini, implementasi PDK SkyWater disederhanakan hanya pada 5 layer metal dan dan via, mengasumsikan bahwa peletakan *standard cell* telah dilakukan sebelumnya.

II. PEKERJAAN TERKAIT

Maze routing yang dikemukakan oleh Lee[2] menelusuri grid 2 dimensi menggunakan ekspansi yang pada hakikatnya adalah BFS. Menjamin bahwa hasil dari jalur yang diberikan adalah jalur terpendek namun akan boros memori dan waktu karena menelusuri rute secara menyebar terlebih pada kanvas besar. Hadlock dan Soukup mengusulkan pencarian terarah atau yang sering dikenal sebagai metode heuristik untuk mengurangi node

yang harus ditelusuri. Metode ini diterapkan pada algoritma pencarian rute tercepat A* untuk memandu pencarian menuju titik target dengan harapan mempercepat pencarian[3].

III. Teori Dasar

A. *Routing* sebagai Pencarian pada *Graf*

Pendekatan klasik yang dikemukakan Lee untuk melakukan routing satu net merupakan *maze-routing*[2], yakni memetakan kanvas routing menjadi grid lalu mencari lintasan menuju tujuan. Setiap sel dapat direpresentasikan sebagai graf dimana semua sel yang bertetangga menjadi sisinya. Sehingga menggunakan konsep dasar ini menjadikan permasalahan routing disederhanakan menjadi lintasan tercepat dalam graf[1]. Pada rangakaian dengan lapisan non-tunggal meningkatkan kompleksitas dari grid dari 2 dimensi menjadi 3 dimensi. Penyederhanaan abstraksi masalah ini menjadikan algoritma pencarian jalur tercepat seperti A* dan BFS dapat diaplikasikan.

B. SKY130 PDK

Dalam produksi sirkuit terpadu, perlu untuk mematuhi aturan desain yang ditetapkan oleh fabrikator. Isi dari aturan desin ini memastikan bahwa desain yang dibuat terjamin untuk dapat difabrikasi. Pada makalah ini menggunakan PDK (*Process Design Kit*) SKY130. SKY130 merupakan PDK proses 130nm dari SKyWater yang dirilis secara *open-source*. Aturan desain ini dapat diakses publik untuk tujuan penelitian dan pendidikan[4]. Pada makala ini tidak menggunakan sepenuhnya aturan dari SKY130, namun menggunakan pemodelan yang disederhanakan.

Pada SKY130 terdapat banyak sekali layer dan via penghubung. Pada makalah ini, diambil 9 layer yang terdiri atas metal 1-5, dan via 1-4. Mengasumsikan bahwa lapisan penyusun MOSFET *local interconnect* merupakan bagian dari *standard cell*. Kemudian konstrain lain yang diterapkan adalah minimum luasan dari tiap metal dan via. Pada PDK SKY130, nilai ini bukanlah bilangan bulat, namun panjang dalam satuan nanometer. Ukuran ini berbeda untuk tiap layer dan semakin membesar untuk tiap layer diatasnya.

TABLE I. Ukuran Minimum Metal

Nama Layer	Deskripsi	*Min Width*
li1	Local Interconnect (TiN)	0,17 µm
met1	Metal 1 (Thin Metal)	0.14 µm
met2	Metal 2 (Thin Metal)	0.14 µm
met3	Metal 3 (Thick Metal)	0.30 µm
met4	Metal 4 (Thick Metal)	0.30 µm
met5	Metal 5 (Top Thick Metal)	1.6 µm

TABLE II. Ukuran Minimum Metal

Nama Layer	Deskripsi	*Min Width*
mcon	Polysilicon / Diffusion ke li1	0,17 µm
via	li1 ke met1	0.15 µm
via2	met1 ke met2	0.20 µm
via3	met2 ke met3	0.20 µm
via4	met3 ke met4	0.20 µm
via5	met4 ke met5	0.80 µm

Note: Pada implementasi ini terdapat sedikit penamaan dimana via1 pada implementasi merujuk indeks mulai via2 / penghubung met1 ke met 2.

C. Algoritma *Breadth-First Search*

BFS merupakan algoritma penarian jalur terpendek dengan menjelajahi graf secara melebar. Semua simpul pada kedalaman tertentu dari titik awal dikunjungi secara merata sebelum beranjak ke kedalaman berikutnya menggunakan struktu antrian FIFO[5].

TABLE III. Pseudocode BFS

Algorithm 1 BFS untuk 1 net

```
Inisialisasi antrian Q yang merupakan seluruh area kontak
While Q tidak kosong do
  U <- node sekitar Q yang ditinjau
  If U anggota area tujuan
    Return rekonstruksi path
  End if
  For all V merupakan anggota U passable do
    If V belum dikunjungi do
      Masukkan V kedalam Q
    End if
  End for
End while
Return gagal
```

D. Algoritma A*

A* merupakan algoritma pencaraian yang memiliki heuristik dengan memberikan value pada simpul mengikuti persamaan $f(n) = g(n) + h(n)$, dengan $g(n)$merupajan biaya perjalanan dan $h(n)$ adalah estimasi heuristik [3]. Simpul dengan f terkecil diekspansi lebih dulu menggunakan *priority queue*.

TABLE IV. Pseudocode A*

Algorithm 1 A* untuk 1 net

```
Untuk tiap sel awal, masukkan ke heap
While tidak null do
  U <- pop min head
  If U tidak bisa ekspansi then
    lanjut
  End if
  Tandai U sudah final
```

8: **If** U anggota tujuan **then**
9: **Return** rekonstruksi jalur perjalanan
10: **End if**
11: **For all** tetangga U yang passable **do**
12: Hitung f dan masukkan ke heap
13: **End for**
14: **End while**
15: **Return** gagal

IV. IMPLEMENTASI DAN METODOLOGI

A. Pemodelan *Grid* dan *Cost*

Kanvas dimodelkan sebagai grid 3 dimensi berukuran W x H x Z. W dan H merupakan area planar yang didefinisikan berukuran 360x360 pada pengujian. Z bernilai tetap Z=9 dengan rincian lapisan sebagai berikut:

Metal 5
Via 4
Metal 4
Via 3
Metal 3
Via 2
Metal 2
Via 1
Metal 1

Fig. 1. Visualisasi Lapisan *Wafer*

Untuk nilai rasio perbandingan ukuran minimum dan jarak minimum diskalakan ulang agar mempermudah pemodelan dimana skala 1 dianggap 1 cell dalam grid yang dibangun pada luasan 2 dimensi W x H. Nilai rasio ini dilakukan pembulatan pada nilai bilangan bulat terdekat dengan rincian pada tabel berikut.

TABLE V. RASIO UKURAN DAN JARAK MINIMUM

Nama Layer	Min Width	Min Spacing
metal1	7	7
metal2	7	7
metal3	15	15
metal4	15	15
metal5	40	80
via1	7	9
via2	12	10
via3	12	10
via4	19	40

Untuk memaksimalkan kepadatan layer terbawah maka untuk penggunaan via akan dikenakan *cost* dengan rincian sebagai berikut:

TABLE VI. TABEL *COST* VIA

Untuk memaksimalkan kepadatan layer terbawah maka untuk penggunaan via akan dikenakan *cost* dengan rincian sebagai berikut:

	via1	via2	via3	via4
cost	10	20	30	40

Untuk persamaan heuristik yang digunakan pada A* menggunakan persamaan manhattan dengan persamaan berikut:

$$h(n) = |x_n - x_g| + |y_n - y_g| + \left[\frac{|z_n - z_g|}{2}\right] \cdot C_{min}$$

Persamaan heuristik ini menghitung jarak pada bidang W x H kemudian menambahkan bobot biaya perpindahan via berdasarkan bobot biaya terendah.

B. Algoritma *Auto-Router*

Pengujian bekerja dengan melakukan pembacaan file input dengan format khusus. Format file input terdiri atas 2 tipe yakni format *netlist* dan format *blockade*. Secara berurut format input antara lain sebagai berikut:

$$NET\ m1, m2, x11, x12, y11, y12, x21, x22, y21, y22$$
$$OBSM/V\ , idx, xmin, xmax, ymin, y\ \max$$

Hasil pembacaan input akan melalui penyusunan tipe data pada program. Kemudian dilakukan penomoran nodal yang akan menomori semua semua netlist yang saling terhubung dalam 1 indeks nodal yang sama. Langkah ini dilakukan untuk meningkatkan efektivitas routing dimana ara awala dan area tujuan semakin luas dan memungkinkan koneksi dari trace yang berdekatan.

Kemudian dilakukan pengecekan semua *obstacle*. *Obstacle* pada setiap layer mengikuti aturan jarak minimum. Sehingga dilakukan pelabelan area yang tidak mungkin dicek oleh algoritma pencarian jalur. Selain itu dilakukan pengecekan cell apakah memenuhi luasan minimum agar dapat diekspansi. Langkah ini mengeliminasi area sempit yang tidak mungkin dilakukan ekspansi untuk membentuk trace yang memenuhi lebar minimum.

Algoritma pencarian jalur bekerja dengan membuat jalur dengan lebar satu cell terlebih dahulu. Kemudian apabila jalur ditemukan pada tiap iterasi maka jalur tersebut diperluas hingga memenuhi aturan lebar minimum. Pada iterasi berikutnya luasan jalur trace yang telah diekspansi menjadi konsiderasi titik yang dapat dihubungkan.

C. Penyetaraan Perbandingan

Untuk menjamin kesetaraan variabel kontrol, kedua algoritma baik BFS maupun A* dijalankan pada dua salinan grid yang identik. Keduanya dikenakan DRC, fungsi tetangga, definisi biaya, serta definisi titik awal dan tujuan yang sama. Dengan demikian diharapkan metrik penilaian berasal sepenuhnya dari performa algoritma.

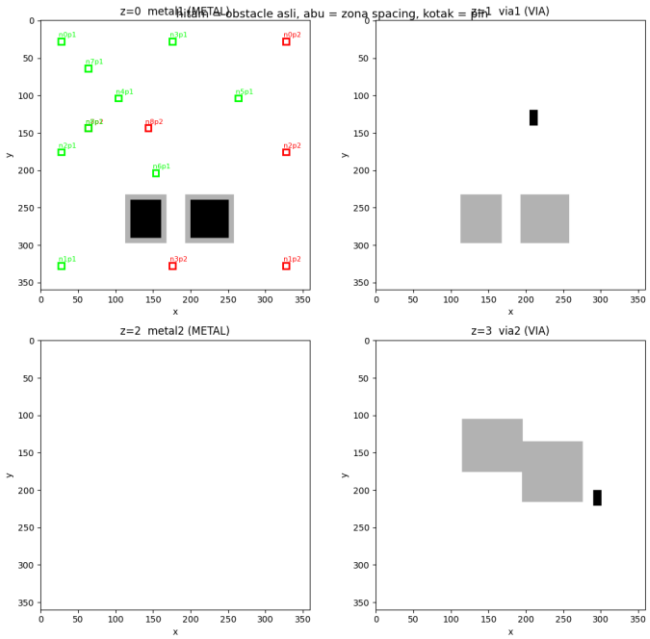
D. Kasus Uji

Kasus Uji berupa 9 net dengan enam obstacle pada grid berukuran 360x360 dengan 9 lapisan. Net dirancang mencangkup kondisi berbagai skenario seperti koneksi antar metal1, koneksi metal1 ke metal5, dan node bercabang. Obstacle ditempatkan pada lapisan logam dan via untuk menguji jalur memutar. Metrik yang dicatat adalah keberhasilan, biaya total lintasan, panjang lintasan, jumlah via, jumlah simpul yang dibuka (nodes expanded), ukuran frontiermaksimum, dan waktu eksekusi.

V. Implementasi dan Metodologi

A. Hasil Eksperimen

Hasil eksperimen dalam tiap tahapan divisualisasikan dalam figur berikut:



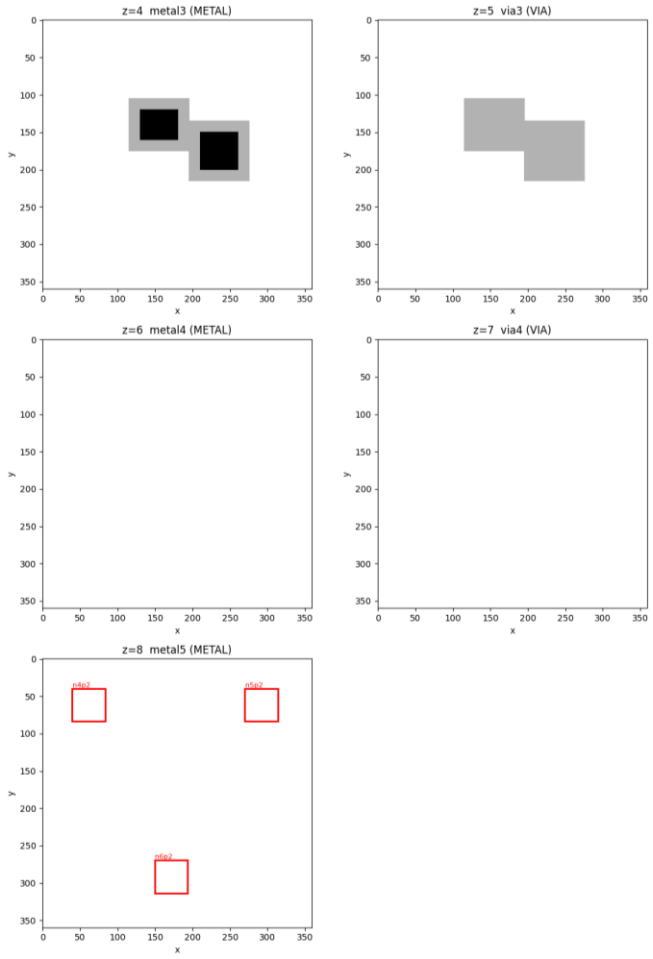
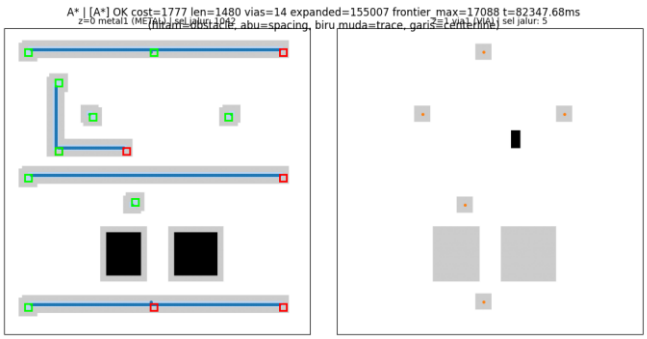


Fig. 2. Visualisasi Kondisi Awal per Layer dengan Titik Kontak Awal di Metal 1 dan Metal 5 serta Penghalang di Metal 1, Via 2, Via 3, dan Metal 4.

Untuk hasil pengujian menggunakan algoritma A* didapatkan hasil sebagai berikut:



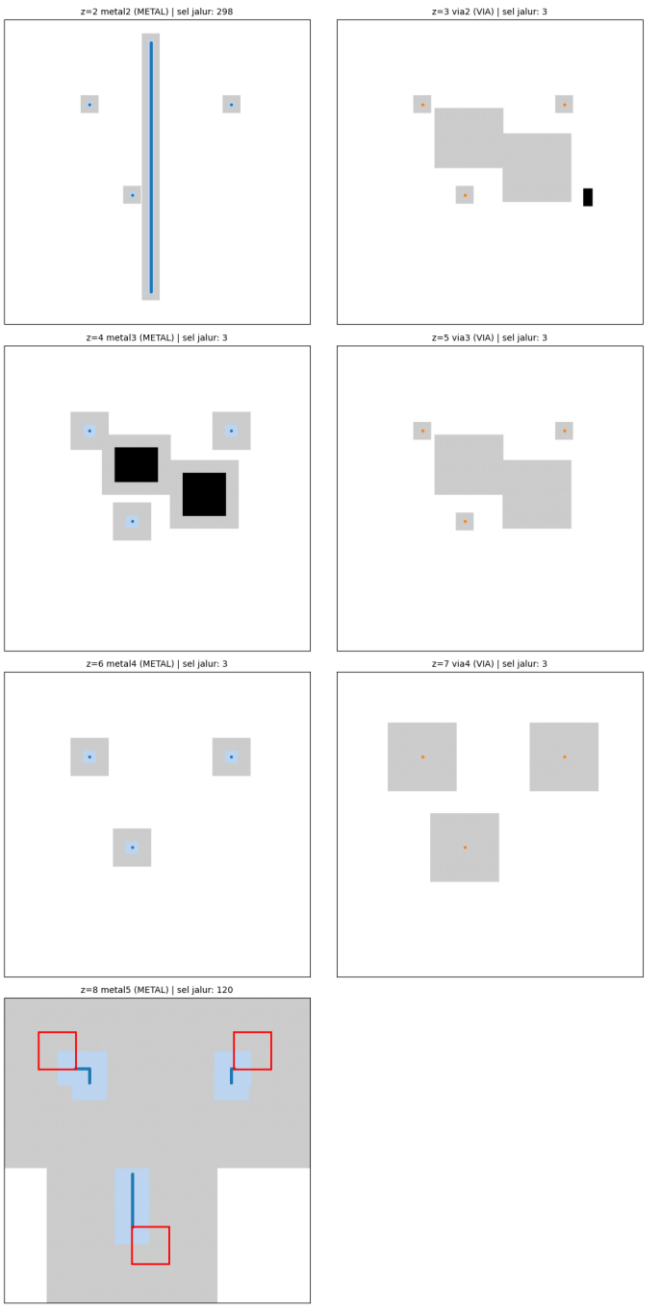
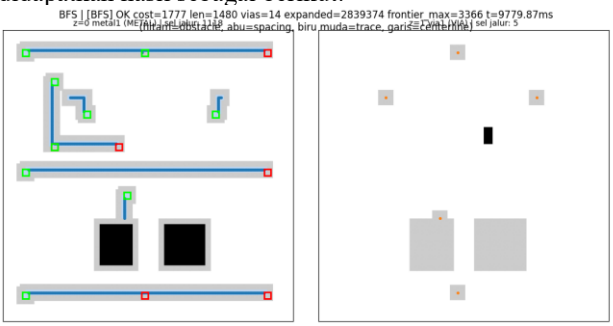


Fig. 3. Visualisasi Hasil Auto-Routing Menggunakan Algortima A*

Untuk hasil pengujian menggunakan algoritma BFS didapatkan hasil sebagai berikut:



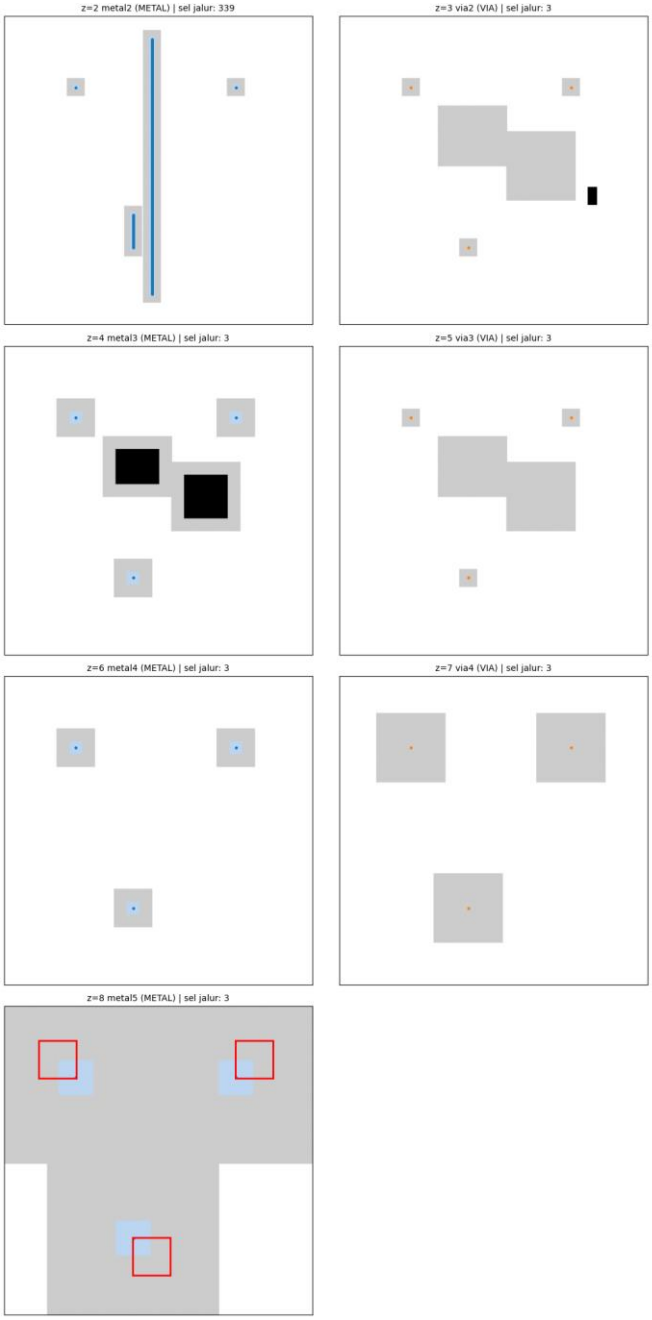


Fig. 4. Visualisasi Hasil Auto-Routing Menggunakan Algortima BFS

Berdasarkan nilai metrik statistik *solver* dari tahapan iterasi algortima diuraikan dalam tabel berikut:

TABLE VII. PERBANDINGAN BFS DAN A* PADA KASUS UJI (9 NET, GRID 360 × 360 × 9)

Metrik	**BFS**	***A****
Biaya total (cost)	1777	1777
Panjang lintasan (sel)	1480	1480
Jumlah via	14	14
Simpul dibuka	2.839.372	155.007

Frontier maksimum	3.366	17.088
Waktu eksekusi (ms)	10.937	100.916

TABLE VIII. SIMPUL DIBUKA PER SEGMEN ROUTING

Seg	Jenis	BFS	*A**	Biaya
1	metal1–metal1	524.341	2.345	293
2	metal1–metal1	534.580	2.345	293
3	metal1–metal1	702.855	2.345	293
4	metal1–metal1	710.518	10.805	319
5	metal1–metal5	33.950	47.334	138
6	metal1–metal5	16.779	38.084	124
7	metal1–metal5	79.959	50.855	167
8	cabang multi-pin	76.388	585	73
9	cabang multi-pin	160.004	309	77
Total		2.839.374	155.007	1.777

B. Kualitas Solusi

Kedua algoritma menghasilkan lintasan dengan biaya total, panjang, dan jumlah via yang identik di skor 1.777, 1.480 sel, dan 14 via. Hasil dari kedua lagortima ini membuktikan bahwa kedua algoritma memastikan hasil merupakan rute optimal dan keunggulan berada pada seberapa optimal dan efidien proses pencarian.

C. Efisiensi Eksplorasi

Perbedaan paling mencolok ada pada jumlah simpul yang dibuka. A* hanya membuka 155.007 simpul, sekitar 5,5% dari 2.839.374 simpul yang dibuka BFS, atau lebih dari 18 kali lebih hemat. Hal ini menegaskan bahwa metode heuristik berhasil memabngka pencarian yang mengarah tidak pada tujuan optimal.

D. Penggunaan Memori dan *Frontier*

Menariknya, frontier maksimum A* (17.088) justru jauh lebih besar daripada BFS (3.366), sekitar lima kali lipat. Penyebabnya adalah sifat best-first: A* menyimpan banyak simpul kandidat dengan nilai f beragam pada antrian prioritasnya, termasuk simpul yang dimasukkan ulang ke heap ketika ditemukan jalur lebih murah (entri lama dibiarkan dan disaring saat pop).

E. Waktu Eksekusi

Temuan lain yang *counter intuitive* adalah waktu eksekusi yang berbanding terbalik jumlah node yang dikunjungi. Meski membuka 18 kali lebih sedikit simpul, A* justru sekitar sembilan kali lebih lambat (100.916 ms berbanding 10.937 ms). Ini dapat dijelaskan dari biaya per simpul sebenarnya timpang. Pada BFS, memproses satu simpul hanya melibatkan operasi antrian dan pengecekan tetangga yang murah. Pada A*, setiap kali sebuah simpul akan dievaluasi, heuristiknya harus dihitung dan mengiterasi seluruh sel.

VI. KESIMPULAN

Hasil pengujian pada makalah ini menunjukkan bahwa pada eksplorasi ruang algoritma A* lebih unggul dari BFS karena mengunjungi 18 kali lebih sedikit *node*. Namun disisi lain memerlukan memori penyimpana *frontire* yang lebih besar alibat heuristik yang mahal. Namun waktu eksekusi dari A* 9 kali lebih lambat daripada BFS. Hal ini dikarenakan perhitungan heuristik yang mahal. Sehingga dapat disimpulkan bahwa algoritma A* dinyatakan lebih efisien secara waktu jika perhitungan nilai heuristik dirancang murah untuk dihitung.

ACKNOWLEDGMENT

Penulis menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada Dr. Ir. Rinaldi Munir, M.T. dan Dr. Nur Ulfa Maulidevi, S.T., M.Sc. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma, atas ilmu, bimbingan, serta kesempatan yang diberikan untuk mendalami penerapan strategi algoritma melalui penelitian dan implementasi mandiri.

REFERENCES

[1] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, VLSI Physical Design: From Graph Partitioning to Timing Closure. Dordrecht: Springer, 2011.

[2] C. Y. Lee, “An Algorithm for Path Connections and Its Applications,”IRE Transactions on Electronic Computers, vol. EC-10, no. 3, pp. 346–365, 1961.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for theHeuristic Determination of Minimum Cost Paths,” IEEE Transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[4] Google and SkyWater Technology, “SkyWater Open Source PDK (SKY130) Documentation.” [Online]. Tersedia: https://skywater-pdk.readthedocs.io/

[5] E. F. Moore, “The Shortest Path Through a Maze,” in Proceedings of an International Symposium on the Theory of Switching, Part II, Cambridge, MA: Harvard University Press, 1959, pp. 285–292.

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, 19 Juni 2026

Anas Fathurrahman 13222033