CLIQUE MAKSIMAL SEBAGAI KONSEP DASAR PEMBUATAN ALGORITMA CLIQUE-BACK UNTUK MENYELESAIKAN MASALAH N-RATU

Diny Zulkarnaen

Dosen Matematika Fakultas Sains dan Teknologi

dinyzul@gmail.com

ABSTRAK

Masalah *N*-ratu adalah suatu masalah penempatan ratu sebanyak *N* pada papan catur berukuran *NxN* dengan syarat tidak ada dua ratu yang saling menyerang. Banyak pendekatan yang dapat dilakukan untuk memecahkan masalah ini. Pada makalah ini digunakan pendekatan teori graf berdasarkan konsep *clique* maksimal sebagai metode penyelesaian masalah. Metode ini selanjutnya dijadikan dasar untuk membuat algoritma. Agar solusi lebih cepat diperoleh, maka digunakanlah algoritma *backtracking*. Penggabungan antara algoritma yang menggunakan konsep *clique* maksimal dan algoritma *backtracking* tersebut dinamakan algortima *clique-back*. Tidak seperti algoritma pada umumnya yang meletakkan satu-per-satu ratu pada kotak papan catur dan memeriksanya agar memenuhi syarat, algoritma ini justru seolah-oleh menempatkan ratu pada setiap kotak, kemudian mengeliminasinya (sesuai syarat) hingga akhirnya diperoleh solusi (jika ada). Proses eliminasi tersebut menyebabkan solusi yang diperoleh lebih cepat.

Keywords : *N*-ratu, graf, *clique*, algoritma

1. PENDAHULUAN

Masalah *N*-ratu merupakan bentuk umum dari masalah 8-ratu yang muncul pertama kali pada tahun 1948. Masalah 8-ratu ini diperkenalkan oleh pemain catur asal Jerman bernama Max Bezzel. Pada tahun 1950, banyak ahli matematika, termasuk Gauss dan George Cantor, tertarik pada masalah ini dan berusaha untuk memecahkannya. Hingga pada akhirnya teka-teki ini berkembang menjadi masalah *N*-ratu. Dalam waktu empat dekade terakhir, banyak metode yang muncul untuk memecahkan masalah

ini, seperti *search-based algorithm* (contohnya algoritma *backtracking*), generasi permutasi, *divide and conquer*, pemrograman integer, jaringan syaraf dan lain sebagainya.

Pada makalah ini, dibahas metode penyelesaian masalah *N*-ratu menggunakan teori graf berdasarkan konsep *clique* maksimal. Selanjutnya dibuat langkah-langkah atau algoritma yang efisien guna memperoleh solusinya. Algoritma yang dimaksud adalah algoritma *clique-back*. Algoritma ini menggunakan gabungan dari konsep *clique* maksimal dan juga *backtracking*.

2. MASALAH *N*-RATU

Masalah *N*-ratu merupakan suatu masalah penempatan *N* buah ratu pada *NxN* papan catur sedemikian sehingga tidak ada dua ratu yang dapat menyerang satu sama lain. Jadi cara penempatan ratu pada papan catur harus memenuhi syarat sebagai berikut:

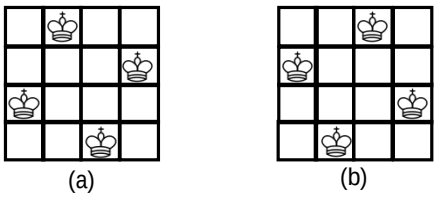
a. Tepat *N* buah ratu ditempatkan pada papan catur
b. Setiap baris hanya memuat sebuah ratu
c. Setiap kolom hanya memuat sebuah ratu
d. Setiap diagonal memuat tidak lebih dari sebuah ratu

Apabila dilakukan pemeriksaan pada seluruh kemungkinan susunan *N* buah ratu pada *M = N*N* kotak yang tersedia, maka solusi yang diperoleh akan membutuhkan waktu yang cukup lama. Banyaknya kemungkinan susunan tersebut adalah ${}_{M}C_{N}$ cara. Misalkan banyaknya kemungkinan menyusun delapan ratu pada 8*8 = 64 kotak adalah ${}_{64}C_{8}$ = 4.426.165.368 cara.

Akan tetapi dapat dimungkinkan untuk mereduksi banyaknya kemungkinan susunan tersebut menjadi N^N, dengan catatan setiap baris hanya diisi oleh sebuah ratu. Jadi banyaknya

kemungkinan susunan dapat dikurangi menjadi 8^8 = 16,777,216 cara. Atau dapat juga direduksi lagi menjadi *N*!, di mana tiap baris dan tiap kolom hanya diisi oleh sebuah ratu. Untuk *N* = 1, hanya terdapat 1! kemungkinan susunan penempatan ratu. Pada kasus ini diperoleh sebuah solusi yang trivial. Sedangkan untuk *N* = 2 dan *N* = 3, masing-masing terdapat 2! = 2 dan 3! = 6 kemungkinan susunan penempatan ratu. Cukup mudah untuk memeriksa seluruh kemungkinan tersebut hingga dapat disimpulkan bahwa tidak ada solusi pada kasus *N* = 2 dan *N* = 3.

Gambar 1 menyajikan dua buah solusi dari masalah 4-ratu. Akan tetapi dua solusi tersebut merupakan solusi yang isomorfis. Hal ini dikarenakan solusi pada Gambar 1b dapat dibentuk dari refleksi terhadap solusi pada Gambar 1a. Jadi, apabila solusi s_1 dapat disusun dari operasi (rotasi atau refleksi) solusi lain s_2, maka solusi s_1 dan s_2 tersebut adalah solusi yang isomorfis dan dihitung sebagai satu solusi yang unik.



Gambar 1. Dua buah solusi yang isomorfis pada masalah 4-ratu

Pada Tabel 1 ditampilkan solusi total (tanpa memperhatikan operasi) dan solusi unik (memperhatikan operasi) dari permasalahan *N*-ratu [4]. Dapat dilihat bahwa semakin besar *N*, semakin banyak solusi yang diperoleh kecuali untuk *N* = 6

N	Solusi Total	Solusi Unik
4	2	1
5	10	2
6	4	1
7	40	6
8	92	12
9	352	46
10	724	92

11	2.680	34
12	14.200	1.787

Tabel 1. Banyaknya solusi untuk masalah *N*-ratu

3. METODE *CLIQUE* MAKSIMAL

Pada makalah ini dibahas metode penyelesaian masalah menggunakan teori graf berdasarkan konsep *clique* maksimal.

Graf Komplemen dan *Clique*

Misalkan terdapat graf $G(V,E)$ dengan n buah simpul V dan m buah busur E, maka dimungkinkan untuk membuat graf $\overline{G}$ dengan n buah simpul. Graf $\overline{G}$ tersebut dinamakan komplemen dari graf G. Untuk setiap pasangan $(u,v) \in E(G)$, maka $(u,v) \notin E(\overline{G})$, begitu pula sebaliknya, untuk setiap pasangan $(u,v) \notin E(G)$, maka $(u,v) \in E(\overline{G})$. Gambar 2 menampilkan suatu graf dengan 4 simpul dan 4 busur beserta graf komplemennya.

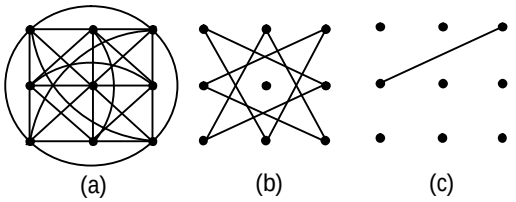
Pada Gambar 2a, terdapat himpunan simpul $V'(G) = \{V_2, V_3, V_4\} \subset V(G)$ yang masing-masing saling beradjacent. Himpunan $V'(G)$ tersebut dinamakan ***clique.*** Jadi *clique* dapat diartikan sebagai kumpulan simpul V' di mana $V'(G) \subset V(G)$ ∋ $\forall\, u,v \in V'(G)$ terdapat busur $(u,v) \in E(G)$. Apabila terdapat $\dot{v}$ yang menyebabkan seluruh simpul pada $V' \cup \{\dot{v}\}$ tidak saling beradjacent, maka V' dinamakan ***clique* maksimal.**

Perlu dicatat bahwa suatu graf G dapat memiliki lebih dari satu *clique* maksimal dan seluruh *clique* maksimal pada graf G tersebut tidak selalu memiliki banyak simpul yang sama. Sebagai contoh himpunan simpul $\{v_2, v_3, v_4\}$ dan $\{v_1, v_3\}$ pada gambar 2a merupakan *clique* maksimal pada graf G.

4. APLIKASI *CLIQUE* MAKSIMAL PADA MASALAH *N*-RATU

Setelah dijelaskan mengenai metode *clique* maksimal, selanjutnya metode ini

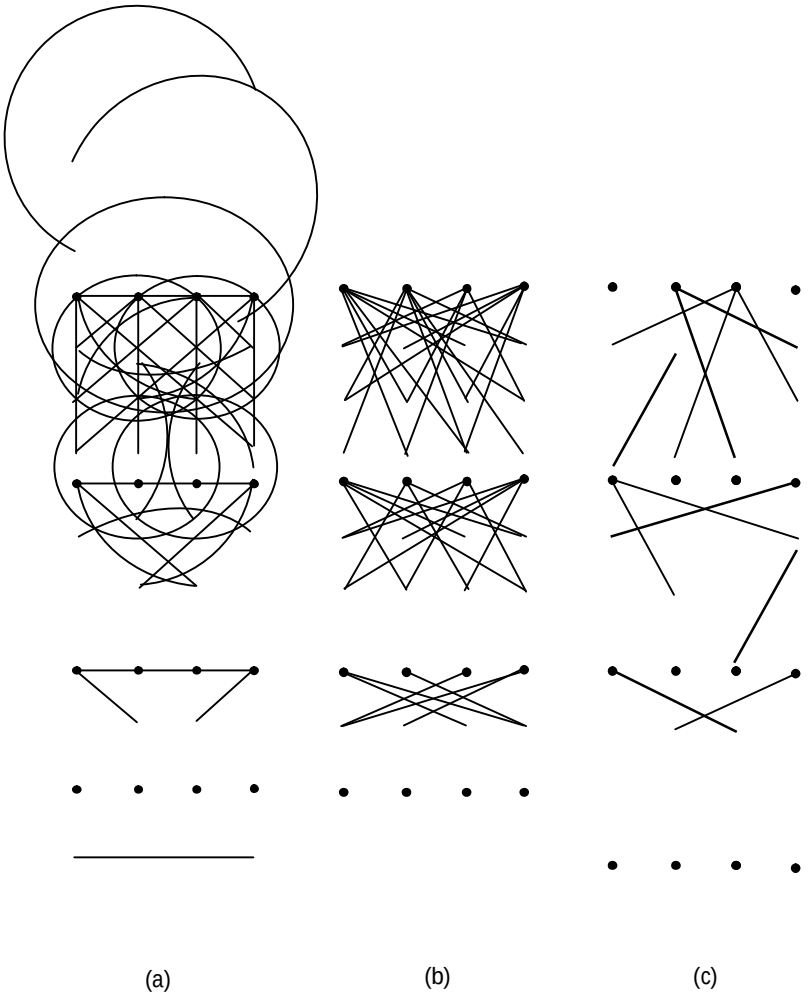
diaplikasikan pada masalah N-ratu. Pertama-tama dipilih sebuah ratu *R* dan dibuat *m* buah simpul di mana *m* = *N***N*. (sebuah simpul merepresentasikan sebuah kotak pada papan catur). Selanjutnya diperiksa setiap kotak *k*, dan cari kotak yang akan diserang apabila *R* ditempatkan pada kotak *k*. Busur pada graf digambarkan berdasarkan aturan : Jika ratu *R* pada kotak *k* dapat menyerang ratu *R*' lain pada kotak *t*, maka terdapat busur yang menghubungkan kotak (simpul) *k* dan *t*.



Gambar 3. (a) Representasi masalah 3-Ratu dalam bentuk Graf *G*. (b) Graf $\overline{G}$ sebagai komplemen dari graf G, (c) salah satu clique maksimal dari $\overline{G}$

Gunakan aturan tersebut pada setiap simpul, sehingga diperoleh graf *G* yang merupakan representasi dari masalah *N*-ratu. Selanjutnya dibuat graf komplemen $\overline{G}$ dari graf G. Jadi busur

ISSN 1979-8911



Gambar 4. (a) Representasi masalah 4-Ratu dalam bentuk Graf *G*. (b) Graf $\overline{G}$ sebagai komplemen dari graf G. (c) Dua buah clique maksimal dari graf $\overline{G}$ yang juga merupakan solusi masalah 4-ratu

pada graf $\overline{G}$ digambarkan berdasarkan aturan ratu pada kotak *k* tidak menyerang ratu lain pada kotak *t*. Selanjutnya dari graf $\overline{G}$ tersebut tentukan *clique* maksimalnya. Solusi diperoleh apabila anggota dari *clique* maksimal tersebut banyaknya adalah *N*. Gambar 3 menunjukkan alur atau

ISSN 1979-8911

proses pencarian solusi masalah 3-ratu menggunakan konsep *clique* maksimal

Banyaknya *clique* maksimal yang diperoleh dari graf $\overline{G}$ adalah delapan buah (salah satunya ditunjukkan pada Gambar 3c) dan diantara *clique* maksimal tersebut tidak ada satupun yang beranggotakan 3 buah simpul. Dengan kata lain untuk masalah 3-ratu tidak diperoleh solusi.

Untuk masalah 4-ratu, alur pencarian solusinya ditunjukkan seperti pada Gambar 4. Pada kasus ini, diperoleh dua buah *clique* maksimal dari graf $\overline{G}$ yang beranggotakan empat buah simpul. Artinya terdapat dua buah solusi dari masalah 4-ratu. *Clique* maksimal yang ditunjukkan pada Gambar 4c tersebut merupakan representasi dari solusi pada papan catur yang telah ditunjukkan sebelumnya pada Gambar 1a dan 1b.

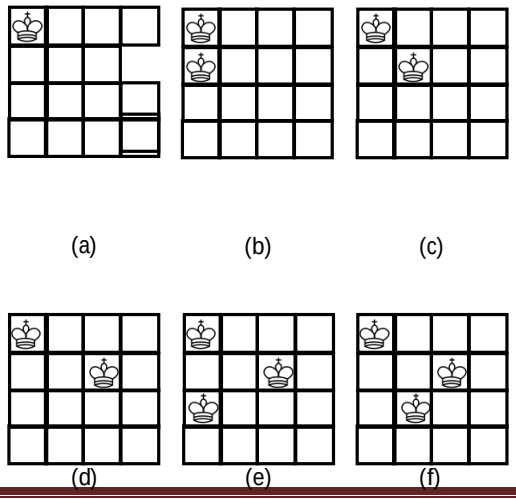
Kesulitan akan ditemui untuk kasus *N* yang besar. Kesulitan yang dimaksud adalah melihat ataupun memeriksa seluruh *clique* maksimal dari graf $\overline{G}$ yang beranggotakan *N* simpul. Maka dari itu, untuk memudahkan pencarian solusi berdasarkan konsep *clique* maksimal, perlu dibuah langkah-langkah (algoritma) yang efisien tanpa harus memeriksanya dari graf.

ISSN 1979-8911

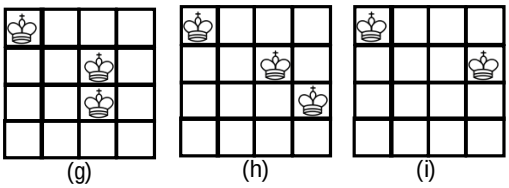
5.1 ALGORITMA *BACKTRACKING*

Algoritma *backtracking* adalah algoritma untuk mencari semua (atau beberapa) solusi terhadap suatu masalah. Cara yang ditempuh dengan membangun kandidat solusi secara bertahap, dan meninggalkan kandidat *c*. Kandidat *c* tersebut hanya sebagai solusi pada tahap tertentu tetapi tidak dapat menghasilkan solusi akhir yang valid.

Aplikasi *backtracking* untuk masalah *N*-ratu adalah sebagai berikut. Pertama-tama letakkan sebuah ratu pada baris pertama dan kolom pertama. Kemudian, letakkan (secara berulang) ratu berikutnya sesuai syarat. Apabila tidak ada kemungkinan untuk meletakkan ratu ke-*i* pada papan catur, maka langkah yang ditempuh adalah mengambil kembali ratu ke-(*i*-1) dan tempatkan ratu ke-(*i*-1) tersebut ke kotak lain yang masih memenuhi syarat. Gambar 7 mengilustrasikan penggunaan algoritma *backtracking*.



ISSN 1979-8911



Gambar 5. Ilustrasi algoritma *backtracking* pada masalah 4-ratu.

Mula-mula sebuah ratu ditempatkan pada baris pertama kolom pertama (Gambar 5a). Kemudian ratu kedua ditempatkan pada baris kedua. Pada baris kedua ini ratu tidak boleh ditempatkan pada kolom pertama (Gambar 5b) dikarenakan dapat diserang oleh ratu pertama, kemudian lakukan penempatan lain yakni pada kolom kedua (Gambar 5c). Disini juga tidak memenuhi syarat, lanjutkan kembali pada kolom ketiga (Gambar 5d). Pada kolom ini ratu kedua tidak dapat diserang oleh ratu pertama. Kemudian lanjutkan penempatan ratu ketiga pada baris ketiga. Oleh karena setiap kolom pada baris ketiga dapat diserang oleh ratu pertama ataupun ratu kedua, maka lakukan *backtracking* dengan cara menempatkan ratu kedua ke kolom yang lain(masih pada baris kedua), yakni pada kolom keempat. Lakukan langkah-langkah tadi secara berulang hingga diperoleh solusi (jika ada).

ISSN 1979-8911

5.2 ALGORITMA *CLIQUE-BACK*

Algoritma *clique-back* menggunakan konsep *clique* maksimal serta *backtracking* untuk memperoleh solusi dari masalah *N*-ratu. Tidak seperti algoritma pada umumnya, yang memulai satu per satu meletakkan ratu pada papan catur dan kemudian memeriksanya agar memenuhi syarat, algoritma ini justru pertama kali yang dilakukan adalah seolah-olah meletakkan ratu di setiap kotak papan catur. Kotak papan catur *NxN* dinyatakan sebagai sebagai kotak k_i, untuk i =1, 2, …, *m* dalam hal ini *m* = *N***N*. Inisialisasi kotak k_i pada masalah 4-ratu ditunjukkan pada Gambar 6. Lebih jelasnya, berikut ini disajikan algoritma *clique-back* untuk memecahkan masalah *N*-ratu.

1. Buat kotak k_i, i = 1, 2, …, *m*. Kotak k_i tersebut menyatakan ratu pada posisi ke i pada papan catur.

   k_1 , …,k_N berada pada baris pertama.

   k_{N+1} , …,k_{2N} berada pada baris kedua.

   N

   $k_{(N-1)\ N\ +\ 1}$, …,k_m berada pada baris ke-*N*.

2. Pilih sebuah kotak pada baris pertama.
3. Eliminasi seluruh kotak yang diserang oleh kotak yang telah dipilih tadi.

   Lakukan langkah (2) dan (3) secara berulang untuk baris kedua hingga baris ke-*N* (jika diperlukan)

4. Apabila dapat dipilih kotak di setiap barisnya, berarti diperoleh

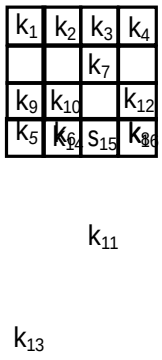
ISSN 1979-8911

solusi. Jika tidak, lakukan backtracking pada baris sebelumnya dan pilih kotak lain.

Berikut ini adalah implementasi dari algoritma *clique-back* untuk masalah 4-ratu dan disajikan dalam bentuk papan catur.

Pertama-tama buat 4*4 = 16 kotak.

$$K = \{k_1, k_2, ..., k_{16}\}$$

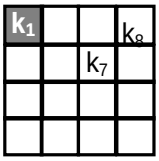


Gambar 6. Enam belas buah kotak pada papan catur berukurang 4x4

Misalkan pada baris pertama dipilih kotak k_1. Akibatnya semua kotak yang diserang oleh k_1 dieliminasi.

Pada baris kedua terdapat dua kemungkinan memilih antara kotak k_7 dan k_8 . Misalkan dipilih simpul k_7. Lalu eliminasikan semua kotak yang diserang oleh k_7.

ISSN 1979-8911

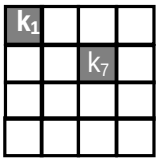


k_{10} k_{12}

k_{14} k_{15}

Gambar 7. Pilih kotak k_1 pada baris pertama

Oleh karena kotak selanjutnya yaitu k_{14} tidak terletak pada baris ketiga, berarti tidak diperoleh solusi.

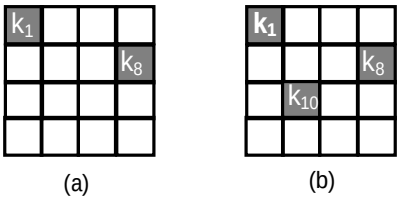


Gambar 8. Pilih simpul k_7 pada baris kedua

k_{14}

Maka dari itu perlu dilakukan *backtracking*, yang artinya kembali pada baris kedua dan dipilih kotak yang lain, yakni k_8. Eliminasikan ratu yang diserang oleh k_8. Selanjutnya satu-satunya kotak yang dapat dipilih adalah kotak k_{10}. Tetapi pemilihan ini pun tidak diperoleh solusi.

 ISSN 1979-8911



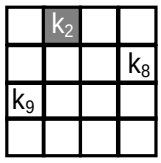
(a) (b)

Gambar 9 (a) Pilih simpul k_8 pada baris kedua

k_{10}

k_{15}

Lakukan *backtracking*, kembali pada baris kedua. Oleh karena tidak ada lagi kotak lain yang dapat dipilih, maka lakukan *backtracking* lagi, kembali pada baris pertama. Misalkan kotak yang dipilih adalah k_2. Lanjutkan dengan mengeliminasi semua kotak yang diserang oleh k_2.



Gambar 10. Pilih simpul k_2 pada baris pertama

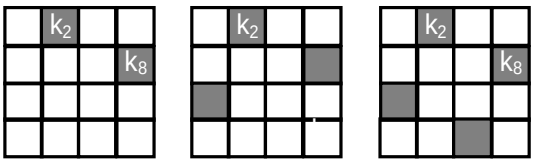
k_{11}

k_{13} k_{15} k_{16}

Kemudian dipilih kotak k_8 pada baris kedua disertai dengan mengeliminasi semua kotak yang diserang k_8. Proses selanjutnya adalah memilih k_9 pada baris ketiga dan mengeliminasi kotak yang diserang k_9. Pada akhirnya dapat dipilih kotak k_{15} pada baris keempat.

ISSN 1979-8911

Oleh karena dapat dipilih *N* buah ratu disetiap barisnya, berarti diperoleh solusi seperti yang ditunjukkan pada Gambar 11c.



k9

k13 k15 k15

(a) (b) (c)

Gambar 11. (a) Pilih simpul k_8 pada baris kedua, (b) pilih simpul k_9 pada baris ketiga dan (c) diperoleh solusi untuk masalah 4-ratu.

Algoritma *clique-back* ini dapat juga dituliskan ke dalam pseudocode sebagai berikut :

Inisialisasi kotak $K \leftarrow [k_1, k_2, k_3,\ldots, k_m]$.

Inisialisasi baris $b \leftarrow 1$

While k_i pada baris ke-b

If $b <$ Panjang (K) then

Pilih k_i

$E \leftarrow$ Eliminasi k_j di mana k_j diserang k_i.

$K \leftarrow K - E$

$b \leftarrow b+1$

Else

* Backtracking $b \leftarrow b - 1$

ISSN 1979-8911

Pilih k_j lain pada baris *b*

If k_j tidak ada then

Lanjutkan ke *

End if

End if

End while

Perhatikan Gambar 12. Pada gambar tersebut menampilkan eksekusi dari pseudocode untuk masalah 4-ratu dengan menggunakan MATLAB. Kotak pada program ini dinotasikan dengan

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Perhatikan bahwa pada baris kedua (pada program) hanya tersisa tiga buah kotak. Padahal untuk masalah 4-ratu harus tersisa 4 ratu yang tidak saling menyerang. Maka dari itu sisa kotak

1 7 14

bukan merupakan solusi. Maka lakukan backtraking dan pilih kotak 8. Dengan menjalankan proses hingga selesai (sesuai prosedur algoritma clique-back) diperoleh solusi

2 8 9 15

yang artinya ratu ditempatkan pada posisi seperti pada Gambar 11c.

 ISSN 1979-8911

```
>> clique
Masukkan ukuran papan catur : 4
     1     7     8    10    12    14    15

     1     7    14

     1     8    10    15

     1     8    10

     2     8     9    11    13    15    16

     2     8     9    13    15

     2     8     9    15
```

Gambar 11. Hasil eksekusi program untuk masalah 4-ratu menggunakan MATLAB.

6. KESIMPULAN

Masalah N-ratu dapat dipecahkan menggunakan berbagai metode, salah satunya adalah dengan menggunakan aplikasi teori graf berupa *clique* maksimal. Akan tetapi kesulitan untuk menemukan solusi ditemui pada saat nilai *N* yang besar. Maka dari itu, untuk memudahkan pencarian solusinya, perlu dibuat langkah-langkah (algoritma) yang efisien tanpa harus memeriksanya dari graf. Algoritma ini dinamakan algoritma *clique-back* yang menggunakan konsep *clique* maksimal sekaligus algoritma *backtracking* untuk memperoleh solusinya.

Langkah pertama yang dilakukan adalah memilih sebuah ratu pada kotak, kemudian mengeliminasi kotak yang diserang oleh ratu yang ditempatkan/dipilih tadi. Jadi kandidat ratu yang akan dipilih selanjutnya

dapat dibuat minimal. Hal ini yang membuat tercapainya hasil (adanya solusi ataupun tidak adanya solusi) lebih cepat. Apabila tidak ditemui solusi, maka dilakukan *backtracking* sebagai usaha untuk mengulang kembali proses pencarian solusinya.

Penulis mengucapkan banyak terima kasih kepada Bapak Djati Kerami yang telah banyak memberikan masukan pada penyusunan makalah ini.

DAFTAR PUSTAKA

[1] Erbas, Cengiz, and Seyed Sarkeshikt and Murat M. Tanik. 1992. *Different Perpective of the N-Queens Problem*, ACM, 99-108

[2] Foulds, L. R. and D. G. Johnson. 1984. *An Application on Graph theory and integer programming*. Mathematics Magazine : vol 52 No 2, 95-104

[3] Mumtaz, Fahmi. 2008. Algoritma Runut-Balik (Backtracking Algorithm) Pada Masalah Knight's Tour. Makalah IF2251 Strategi Algoritmik Tahun 2008.

ISSN 1979-8911