PENGGUNAAN TEKNIK HEURISTIK DAN ALGORITMA RUNUT-BALIK UNTUK PEMECAHAN MASALAH PADA PERMAINAN "HASHIWOKAKERO"

Gisca Tamara

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Infromatika
Institut teknologi Bandung
Jalan Ganesha 10 Bandung
e-mail : if17003@students.if.itb.ac.id

ABSTRAK

Hashiwokakero **adalah jenis permainan teka-teki logika yang dimainkan pada kotak persegi panjang tanpa ukuran standar. Pada kotak ini dapat berisi angka dari 1-8** ***inklusif*** **yang merepresentasikan pulau-pulau dan sisanya adalah kosong (0). Tujuan dari permainan ini adalah menghubungkan semua pulau menjadi terhubung satu grup dengan menggambar serangkaian jembatan antar pulau-pulau.**
Makalah ini membahas mengenai cara untuk menyelesaikan masalah pada permainan ***hashiwokakero*** **baik dengan teknik** ***heuristic*** **maupun dengan algoritma runut-balik. Pendekatan melalui teknik** ***heuristic*** **adalah pendekatan untuk pencarian solusi yang didasarkan pada hal-hal yang sering terjadi pada kasus yang hampir sama, sehingga kita dapat mengeliminasi beberapa kemungkinan solusi tanpa harus mengeksplorasinya lebih lanjut. Sedangkan algoritma runut-balik adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih mangkus dengan tidak memeriksa semua kemungkinan solusi yang ada akan tetapi hanya memeriksa yang mengarah ke solusi saja.**

Kata kunci: *Hashiwokakero, heuristic*, runut-balik, *backtracking*

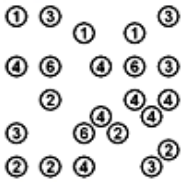
1. PENDAHULUAN

1.1 Peraturan Permainan

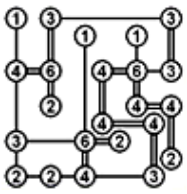
Permainan *hashiwokakero* atau lebih sering disebut dengan *hashi* adalah permainan yang cukup sederhana yang diperkenalkan pertama kali oleh Nikoli. *Hashiwokakero* dapat dimainkan hanya dengan menggunakan kertas kosong dan alat tulis saja. Pada dasarnya permainan ini dimulai dengan mengisi beberapa kotak-kotak kosong dengan angka dari 1-8 dan membiarkan kotak-kotak lain kosong. Setiap angka selanjutnya akan merepresentasikan sebuah pulau yang mempunyai bobot tentang banyak jembatan yang dimiliki.

Tujuan akhir dari permainan ini adalah menghubungkan pulau-pulau tersebut sehingga semua pulau dapat terhubung menjadi satu grup dengan menggambarkan serangkaian jembatan antara pulau-pulau. Akan tetapi jembatan ini harus mengikuti criteria tertentu:

1. Harus dimulai dan berakhir pada pulau-pulau yang berbeda, perjalanan yang berupa sebuah garis lurus di antara pulau-pulau
2. Jembatan yang satu tidak boleh melewati jembatan lain atau pulau (berpotongan)
3. Jembatan yang dibangun hanya boleh pada arah vertikal dan horizontal saja
4. Paling banyak adalah dua jembatan yang menghubungkan antara sepasang pulau
5. Jumlah jembatan yang terhubung ke setiap pulau harus sesuai dengan jumlah (bobot) yang dimiliki oleh pulau tersebut.



Gambar 1. Keadaan awal permainan hashiwokakero



Gambar 2. Keadaan akhir permainan hashiwokakero

1.2 Heuristik

Teknik *heuristic* adalah teknik yang digunakan untuk mengeliminasi beberapa kemungkinan solusi tanpa harus

mengeksplorasinya secara penuh. Selain itu, *heuristic* juga dapat membantu kita untuk memutuskan kemungkinan solusi mana yang pertama kali perlu dievaluasi. *Heuristic* dirancang untuk memecahkan masalah dengan mengabaikan apakah solusi yang dihasilkan dapat dibuktikan secara sistematis benar, tapi seringkali menghasilkan solusi yang bagus.

Heuristic berbeda dengan algoritma karena *heuristic* berlaku sebagai petunjuk, sedangkan algoritma adalah urutan langkah-langkah untuk menyelesaikan suatu masalah. *Heuristic* mungkin tidak selalu memberikan hasil yang diinginkan, tetapi secara ekstrim ia bernilai pada pemecahan masalah. Heuristik yang bagus dapat secara dramatis mengurangi waktu yang dibutuhkan untuk memecahkan masalah dengan cara mengeliminir kebutuhan untuk mempertimbangkan kemungkinan solusi yang tidak perlu. *Heuristic* tidak menjamin selau dapat memecahkan masalah, tetapi seringkali memecahkan masalah dengan cukup baik untuk kebanyakan masalah dan seringkali lebih cepat daripada pencarian solusi secara lengkap.

1.3 Algoritma Runut-Balik

Algoritma runut-balik adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih mangkus. Dengan metode ini, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang selalu dipertimbangkan.

Properti dari metode runut-balik antara lain:

1. Solusi persoalan, yang dinyatakan sebagai vector dengan n-tuple:
   $X = (x_1,x_2,\ldots,x_n)$, $x_i \in$ himpunan berhingga S_i.
2. Fungsi pembangkit nilai X_k, yang dinyatakan sebagai T(k) yaitu membakitkan nilai untuk x_k, yang merupakan komponen vector solusi
3. Fungsi pembatas, yang dinyatakan sebagai $B(x_1,x_2,\ldots x_k)$

Skema umum algoritma runut balik dapat digambarkan melalui dua versi yaitu versi rekursif dan iterative. Berikut adalah pseudo-code dari masing-masing versi.

a. Versi rekursif

```
procedure RunutBalikR(input k : integer)
{
  Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif. Masukan: k, yaitu indeks komponen vector solusi, x[k]. Keluaran: solusi x = (x[1],x[2],…x[k])
}
Algoritma:
for tiap x[k] yang belum dicoba sedemikian sehingga
      (x[k] ← T[(k)) and B(x[1],x[2],…,x[k]) = true do
  if (x[1],x[2],…x[k]) adalah lintasan dari akar ke daun
  then
      CetakSolusi(x)
  endif
  RunutBalikR(k+1) {tentukan nilai untuk x[k+1]}
endfor
```

b. Versi iteratif

```
procedure RunutBalikI(input n : integer)
{
  Mencari semua solusi persoalan dengan metode runut-balik; skema rekursif. Masukan: n, yaitu indeks komponen vector solusi. Keluaran: solusi x = (x[1],x[2],…x[n])
}
Deklarasi:
k : integer

Algoritma:
k ← 1
while k > 0 do
 if (x[k] belum dicoba sedemikian sehingga x[k] ← T(k)
 ) and (B(x[1],x[2],…x[k])=true) then
    if (x[1],x[2],…x[k]) adalah lintasan dari akar ke daun
    then
          CetakSolusi(x)
    endif
    k ← k+1 {indeks anggota tupple berikutnya}
 else
    k ← k-1 {runut-balik ke anggota tupple sebelumnya}
 endif
endwhile
{k = 0}
```

2. PENCARIAN SOLUSI DENGAN TEKNIK HEURISTIK

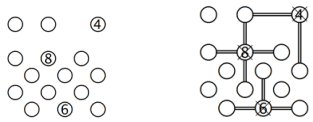
Teknik *heuristic* dapat membantu kita untuk menemukan solusi dari permainan *hashiwokakero*. Meskipun tidak dapat dibuktikan secara matematis seperti halnya algoritma, teknik ini cukup efektif untuk menemukan solusi permasalah dengan lebih cepat. Berikut ini akan dijelaskan beberapa teknik-teknik secara heuritstik yang dapat membantu untuk menyelesaikan permasalahan dari permainan *hashiwokakero*.

2.1 Starting Techniques

Beberapa puzzle *hashiwokakero*, khususnya pada level mudah, tersirat trik untuk memulai permainan pada saat pertama kali melihatnya. Situasi ini terjadi ketika sebuah pulau mempunyai angka maksimum dari jembatan yang diperbolehkan. Pada beberapa puzzle, terdapat pulau-pulau yang tidak mempunyai petunjuk untuk diselesaikan dan tidak relevan dengan teknik penyelesaian pada level tersebut.

- **Pulau dengan bobot 4 di sudut, 6 pada sisi, dan 8 pada posisi tengah**

Pulau yang berada di sudut mempunyai maksimal dua tetangga dan jumlah daru jembatan ke setiap tetangga tidak dapat lebih dari dua. Oleh karena itu, pulau yang berada di sudut dengan bobot 4 pasti mempunyai dua jembatan yang terhubung dengan masing-masing tetangganya. Sama halnya dengan pulau berbobot 6 pada sisi yang hanya mempunyai tetangga maksimal 3 dengan masing-masing tetangga terhubung dengan maksimal dua jembatan.



Gambar 3. Pulau dengan bobot 4 di sudut, 6 pada sisi, dan 8 pada posisi tengah

2.2 Basic Techniques

Langkah selanjutnya adalah menggunakan teknik dasar, dimana kita tidak dapat selalu menyelesaikan semua penghubungan jembatan pada suatu pulau, tetapi kita dapat menentukan keputusan untuk menempatkan satu atau lebih jembatan pada arah tertentu. Berikut ini penggunaan dari teknik dasar.

- **Pulau dengan tetangga tunggal**

Pulau yang memiliki bobot 1 pada baris bawah hanya mempunyai satu tetangga, yang berarti kita harus menghubungkan kedua pulau itu dengan satu jembatan. Perhatikan bahwa tidak ada pulau berbobot 3 atau lebih yang memiliki satu tetangga.

- **Pulau dengan bobot 3 di sudut, 5 pada sisi, dan 7 pada posisi tengah**

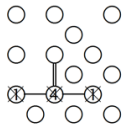
Pulau yang berbobot 3 pasti mempunyai 2 tetangga yang akan dihubungkan dengan 1 jembatan dan 2 jembatan. Tetapi kita tidak tahu pulau mana yang harus dihubungkan dengan 2 jembatan. Yang dapat kita ketahui adalah setiap pulau tetangganya pasti terhubung dengan minimal 1 jembatan. Pada pulau berbobot 5 pasti mempunyai 3 tetangga, dan pulau berbobot 7 pasti mempunyai 4 tetangga.

- **Kasus khusus dari pulau dengan bobot 3 di sudut, 5 pada sisi, dan 7 pada posisi tengah**

Jika pulau di sudut dengan bobot 3 dan tetangganya mempunyai bobot 1, maka semua kondisi akan terpenuhi dan jembatan dapat dihubungkan. Hal yang sama terjadi jika pulau berbobot 5 atau 7 dan tetangga mempunyai bobot 1.

- **Kasus khusus jika pulau dengan bobot 4 berada pada sisi**

Pada contoh dapat dilihat pulau berbobot 4 mempunyai 3 tetangga. Hal ini diperbolehkan selama tidak ada 2 jembatan yang diperbolehkan pada arah yang sama.



Gambar 4. Kasus khusus jika pulau dengan bobot 4 berada pada sisi

- **Kasus khusus jika pulau dengan bobot 6 berada pada posisi tengah**

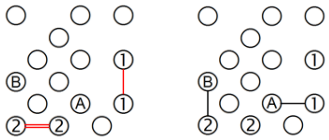
Diasumsikan pulau berbobot 6 terhubung dengan pulau berbobot 1. Hal ini akan meninggalkan 5 jembatan untuk dihubungkan ke tiga tetangga lainnya. Jadi masing-masing tetangga akan terhubung dengan minimal 1 jembatan.

2.3 Isolation Techniques

Salah satu aturan *hashiwokakero* adalah semua pulau harus terhubung dalam 1 jaringan. Aturan ini sangat penting bahkan pada level mudah sekalipun terkadang menemui situasi dimana puzzle tidak dapat diselesaikan. Berikut adalah cara untuk menggunakan isolation techniques.

- **Pengisolasian segmen dua pulau**

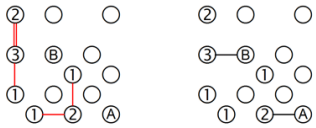
Lihat contoh, jika kita mengubungkan pulau berbobot 1 dengan suatu pulau berbobot 1, maka dua pulau itu akan terisolasi. Penyelesaiannya adalah menghubungkan pulau berbobot 1 dengan pulau A. Begitu pula dengan pulau berbobot 2 yang dihubungkan dengan 2 jembatan pada pulau berbobot 2 yang lainnya.



Gambar 5. Pengisolasian segmen dua pulau

- **Pengisolasian segmen tiga pulau**

Teknik diatas dapat dikembangkan menjadi segmen 3 pulau pualu berbobot 2 pada baris barah tidak dapat dihubungkan seperti contoh karena akan terisolasi. Hal ini berarti kita harus menghubungkan pulau berbobot 2 dengan pulau A.

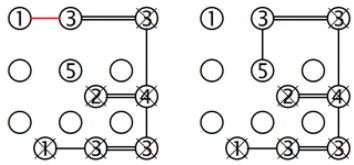


Gambar 6. Pengisolasian segmen tiga pulau

- **Pengisolasian ketika suatu segmen dihubungkan dengan pulau**

Terkadang segmen dari pulau akan terisolasi pada suatu kondisi ketika sulit untuk menemukan jalan dan semangat untuk menyelesaikan. Lihat contoh, ketika pulau dengan bobot 3 masih membutuhkan 1 jembatan lagi. Ketika kita menghubungkan pulau 3 dengan pulau yang

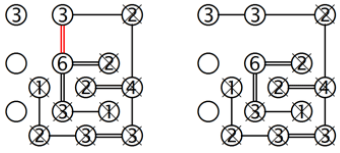
berbobot 1, maka kita akan berakhir pada segmen yang terisolasi, jadi kita seharusnya mengubungkannya dengan pulau yang berbobot 5.



Gambar 7. Pengisolasian ketika suatu segmen dihubungkan dengan pulau

- **Pengisolasian ketika segmen terhubung dengan segmen lainnya**

Pada contoh, terdapat pulau dengan bobot 6 yang kurang untuk menguhubungkan 2 jembatan lagi. Jika kita menghubungkan pulau 6 dengan pulau 3 dengan 2 jembatan, maka kita akan mendapatkan pulau yang terisolasi. Oleh karena itu, kita harus menghubungkan minimal 1 jembatan ke pulau yang berada di sudut kiri atas.



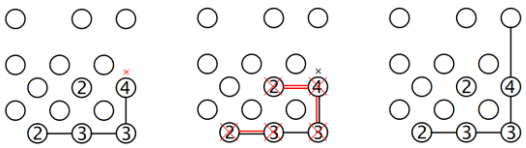
Gambar 8. Pengisolasian ketika segmen terhubung dengan segmen lainnya

2.4 Advanced Techniques

Teknik-teknik dasar tidak akan cukup untuk menyelesaikan pemecahan masalah *hashiwokakero*, oleh karena itu kita membutuhkan advanced techniques untuk kondisi yang khusus dan menarik. Banyak dari teknik lanjut yang menggunakan rekursi, melihat pada proses dari pembentukan asumsi dan memeriksa jika terdapat konflik antara 1 atau 2 step berikutnya.

- **Mengisolasi segmen dengan memblok jembatan**

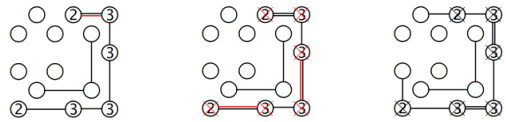
Jika kita mengasumsikan tidak ada jembatan yang dihubungkan pada arah dari X di diagram sebelah kiri maka 5 pulau harus dihubungkan berdasarkan pada diagram yang berada di tengah yang akan menghasilkan segmen yang terisolasi. Oleh karena itu, kita harus menghubungkan minimal 1 jembatan ke atas sebagaimana terlihat pada gambar.



Gambar 9. Mengisolasi segmen dengan memblok jembatan

- **Mengisolasi segmen dengan menambah jembatan**

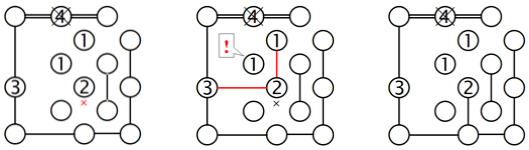
Jika kita mengasumsikan jembatan kedua pada pulau dengan bobot 2 dihubungkan dengan pulau pada sebelah kanannya maka keenam pulau akan terisolasi (diagram tengah). Oleh karena itu, pulau ini harus dihubungkan sebagaimana pada diagram sebelah kanan dan pualu yang berada di sudut bawah harus dihubungkan dengan pulau yang berada diatasnya.



Gambar 10. Mengisolasi segmen dengan menambah jembatan

- **Mengisolasi pulau dengan jembatan**

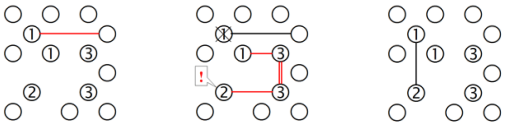
Kita asumsikan bahwa tidak ada jembatan pada arah dari X maka 2 jembatan harus dihubungkan (diagram tengah). Hal ini akan menyebabkan pulau dengan bobot 1 menjadi terisolasi karena pulau yang dapat terhubung dengannya sudah mempunyai jembatan yang lengkap. Oleh karena itu, kita akan menempatkan minimal 1 jembatan kea rah bawah (diagram kanan).



Gambar 11. Mengisolasi pulau dengan jembatan

- **Membuat hubungan jembatan konflik**

Pulau dengan bobot 1 pada baris kedua dapat dihubungkan pada 2 arah. Mari kita asumsikan bahwa pulau ini akan dihubungkan ke pulau yang paling jauh (diagram kiri). Hal ini akan membimbing kita kepada hubungan dari 4 pulau (diagram tengah). Bagaimanapun, pulau dengan bobot 2 masih belum lengkap, terdapat 1 jembatan yang belum terhubung yang akan mengakibatkan konflik. Maka, kita akan menghubungkan pualu dengan bobot 1 kepada pulau yang berada di bawahnya.



Gambar 12. Membuat hubungan jembatan konflik

3. PENCARIAN SOLUSI DENGAN ALGORITMA BACKTRACKING-MULTIGRAF PLANAR BERBOBOT

Dalam permainan *hashiwokakero* kita dapat menggunakan algoritma backtracking mengadopsi pencarian multigraf planar berbobot untuk menemukan solusi dari permainan.

3.1 Pemecahan Masalah

Pencarian solusi pada permainan *hashiwokakero* merupakan pencarian multigraf planar berbobot yang saling terhubung dengan batasan berupa beberapa sisi yang tidak diperbolehkan dan tidak dapat berdampingan.

Struktur data yang digunakan adalah array. Berikut ini akan dijelaskan secara garis besar mengenai elemen-elemen yang mendukung pencarian solusi. Pertama kita representasikan kotak permainan dengan array 2 dimensi yang berisi nilai-nilai dari setiap cell, yaitu $0 \leq x \leq 8$. Jika cell bernilai 0 maka merepresentasikan jalan, jika cell berisi nilai selain 0, maka cell tersebut merepresentasikan pulau dengan bobotnya.

Untuk mencari solusi ini kita akan menggunakan array yang diberi nama *V*. Secara formal, kita diberi suatu set V = {v0,v1,…,vn-1} dari n simpul dimana setiap simpul terdapat bobot D = {d0,d1,…,dn-1}. Array *V* akan berisi simpul-simpul yang ada dalam permainan.

Untuk pilihan, kita dapat memberikan suatu set sisi *Eh* = {{vi,vj} | i ≠ j} merepresentasikan sisi horizontal yang diperbolehkan atau dibutuhkan pada suatu graf dan *Ev* = {{vi,vj} | i ≠ j} merepresentasikan sisi vertikal yang diperbolehkan atau dibutuhkan pada suatu graf. Sehingga terdapat $E = Eh \cup Ev$. Tujuan kita adalah mencari prosedur yang memastikan untuk setiap graf sederhana G = (V,E), dibuat dengan menambahkan sisi-sisi E diantara simpul pada V, G terhubung.

```
procedure EdgeH (input V1,V2,V3 : vertex)
{sisi horizontal yang memungkinkan untuk dihubungkan}
if x.V1 = x.V2 and y.V1 < y.V2 then
x.V3 ← x.V1
for y.V3 > y.V1 to y.V3 < y.V2 do
    Eh[x][y] ← 0
endif
endfor
```

```
procedure EdgeV (input V1,V2,V3 : vertex)
{sisi vertikal yang memungkinkan untuk dihubungkan}
if y.V1 = y.V2 and x.V1 < x.V2 then
xyV3 ← y.V1
for x.V3 > x.V1 to x.V3 < x.V2 do
    Ev[x][y] ← 0
endif
endfor
```

Terdapat indikator yang digunakan untuk mengetahui suatu sisi antar pulau telah dipakai satu kali atau dua kali dan jumlah sisi yang ada pada satu pulau harus sesuai dengan bobot yang dimiliki oleh pulau tersebut. Kita juga harus mengecek apakah ada sisi yang saling berpotongan ketika kita akan menghubungkan dua buah pulau. Jika ada sisi-sisi yang berpotongan maka pembuatan jembatan antar pulau tersebut tidak valid dan harus mencari pulau lainnya.

```
procedure noCross1 (input Eh,Ev : edge)
{pengecekan tidak adanya perpotongan antara sisi-sisi}
Deklarasi:
xe1(E), xe2(E) : integer {diisi oleh biner 0/1}
Algoritma:
if Ev(x1) < Eh(x) and Ev(x2) > Eh(x) and Eh(y1) < Ev(y)
and Eh(y2) > Ev(y)
xe1(Eh) + xe1(Ev) <= 1
endif
```

```
procedure noCross2 (input Eh,Ev : edge)
{pengecekan tidak adanya perpotongan antara sisi-sisi}
Deklarasi:
xe1(E), xe2(E) : integer {diisi oleh biner 0/1}
Algoritma:
if Ev(x1) < Eh(x) and Ev(x2) > Eh(x) and Eh(y1) < Ev(y)
and Eh(y2) > Ev(y)
xe1(Eh) + xe2(Ev) <= 1
endif
```

```
procedure noCross3 (input Eh,Ev : edge)
{pengecekan tidak adanya perpotongan antara sisi-sisi}
Deklarasi:
xe1(E), xe2(E) : integer {diisi oleh biner 0/1}
Algoritma:
if Ev(x1) < Eh(x) and Ev(x2) > Eh(x) and Eh(y1) < Ev(y)
and Eh(y2) > Ev(y)
xe2(Eh) + xe1(Ev) <= 1
endif
```

```
procedure noCross4 (input Eh,Ev : edge)
{pengecekan tidak adanya perpotongan antara sisi-sisi}
Deklarasi:
xe1(E), xe2(E) : integer {diisi oleh biner 0/1}
Algoritma:
if Ev(x1) < Eh(x) and Ev(x2) > Eh(x) and Eh(y1) < Ev(y)
and Eh(y2) > Ev(y)
xe2(Eh) + xe2(Ev) <= 1
endif
```

Ada pula simpul target dan simpul sumber yang akan dihubungkan. Simpul target haruslah mempunyai nilai baris atau kolom yang sama dengan simpul sumber dan memastikan antar simpul sumber dan simpul target tidak ada pulau lain.

Kita akan menggunakan prinsip dari aliran jaringan dimana ketika suatu simpul dijadikan target, maka simpul lainnya akan mengirimkan bobotnya (jumlah sisi yang dapat dihubungkan) kepada simpul target. Indikator sisi yang tadi didefinisikan berperan sebagai *variable upper bound* dimana akan menentukan pembentukan graf yang memungkinkan yang didefinisikan sebagai fungsi pembatas dalam pembentukan graf. Jika jumlah sisi yang diberikan oleh simpul target kepada simpul sumber lebih besar dari sisi yang disediakan oleh simpul sumber, maka hubungan menjadi tidak valid dan harus mencari simpul-simpul lainnya yang sesuai.

Dari elemen-elemen yang mendukung pencarian solusi, kita dapat menentukan bahwa solusi permasalahan adalah X = (x_1,x_2,…,x_n), $x_i \in \{(A_{i,j},E_{i,j}), (A_{i+1,j+1},E_{i+1,j+1}), \ldots, (A_{n,n},E_{n,n})\}$ dengan fungsi pembatas adalah $\sum_{j=0}^{n=1} Ai,j = di$ dan *variable upper bound*. Dengan jembatan yang terhubung pada suatu pulau ditentukan dengan algoritma berikut:

1. Bangkitkan simpul V_i
2. Periksa tetangga yang memungkinkan melakukan hubungan
3. Jika memenuhi semua batasan yang ada sesuai dengan elemen-elemen pendukung pencari solusi, maka hubungkan kedua pulau tersebut. Ulangi langkah (2) untuk tetangga berikutnya.
4. Jika tidak memenuhi batasan yang ada, maka ulangi langkah (2) untuk tetangga berikutnya.
5. Jika tidak ada lagi tetangga yang memungkinkan melakukan hubungan, maka maju ke simpul berikutnya
6. Jika tidak ada lagi simpul yang dibangkitkan, maka persoalan pencarian solusi *hashiwokakero* berakhir.

4. KESIMPULAN

Kesimpulan yang dapat diambil dari analisis permainan *hashiwokakero* baik menggunakan teknik *heuristic* dan algoritma runut-balik adalah:

1. Teknik *heuristic* dapat memecahkan masalah pencarian solusi dengan lebih cepat akan tetapi tidak dapat dibuktikan secara matematis. Banyak teknik heuristk yang dapat membantu kita untuk menemukan solusi pada permainan *hashiwokakero*. Pada kenyataannya teknik *heuristic* sering membantu untuk mengeliminasi pencarian yang tidak mengarah ke solusi.
2. Pencarian solusi dengan algoritma runut-balik menghasilkan solusi yang lebih mangkus daripada dengan algoritma DFS.
3. Pada kenyataannya, pemain biasanya menggunakan penggabungan antara teknik *heuristic* dan algoritma runut-balik untuk memecahkan permasalah pada permainan *hashiwokakero*.

REFERENSI

[1] Nikoli Co., Ltd.: Puzzle Cyclopedia. 2004. ISBN 4-89072-406-0.

[2] Munir, Rinaldi. "Diktat Kuliah IF3051 Strategi Algoritma". Program Studi Teknik Informatika ITB. 2009.

[3] Daniel Andersson, "HASHIWOKAKERO Is NP-Complete", 2.

[4] http://en.wikipedia.org/wiki/Hashiwokakero. Diakses pada 10 Desember 2009 pukul 13.00

[5] http://www.nikoli.com/en/puzzles/hashiwokakero/. Diakses pada 10 Desember 2009 pukul 15.00

[6] http://www.conceptispuzzles.com/index.aspx?uri=puzzle/hashi/techniques. Diakses pada 10 Desember 2009 pukul 11.00

[7] http://www.indigopuzzles.com/ipuz/help.action?helpId=hashi/index. Diakses pada 12 Desember 2009 pukul 11.00

[8] http://www.sudokutiger.com/Hint_Hashi.htm. Diakses pada 12 Desember 2009 pukul 13.00