Jurnal Algoritme
Vol. 1, No. 1, Oktober 2020, Hal. 12 – 20

Perbandingan Algoritma *Dijkstra* dan Algoritma A *Star* Pada Permainan Pac-Man

Ahmad Wildan Rizky Ramadhan *[1], Daniel Udjulawa [2]
[1,2]STMIK GI MDP; Jalan Rajawali No. 14 Palembang, Telp : (0711)376400
e-mail: *[1]awildanrr@mhs.mdp.ac.id, [2] daniel@mdp.ac.id

Abstrak

AI (Artificial Inteligence) atau yang disebut juga dengan kecerdasan buatan merupakan salah satu cabang dari ilmu komputer untuk memberikan suatu pengetahuan pada komputer agar dapat mampu menyelesaikan tugas – tugas atau berpikir seperti manusia. Salah satu contoh kecerdasan buatan yang dapat diterapkan pada game adalah Path Finding. Path Finding adalah salah satu kecerdasan buatan yang dipakai untuk menentukan jaur terpendek antara titik awal dengan titik akhir. Logika Fuzzy merupakan ilmu yang mempelajari mengenai ketidakpastian. Logika Fuzzy juga mampu untuk memetakan suatu ruang input kedalam suatu ruang output dengan tepat. Metode yang digunakan dalam penelitian ini adalah meode prototype dimana tahap-tahap yang dilakukan adalah menganalisis kebutuhan, mendesain prototype, implementasi, dan pengujian. Tujuan utama yang ingin dicapai dari penelitian ini adalah Untuk membandingkan performa algoritma Djikstra dan algoritma A Star untuk penyelesaian game Pac-Man. Hasil yang didapatkan untuk algoritma Dijkstra adalah 2 kali gagal, dan 1 kali berhasil dalam menyelesaikan permainan dengan score 4100, 3350, 3940, sedangkan untuk algoritma A Star mendapatkan hasil 2 kali berhasil, dan 1 kali gagal dengan score 4300, 2350, 3450. Dari kedua Algoritma yang digunakan untuk menyelesaikan permaian PAC-MAN dengan mendapatkan score terbaik adalah algoritma A Star.

Kata kunci: Path Finding, Dijkstra, A Star, fuzzy logic, Artificial Inteligence

Abstract

AI (Artificial Intelligence) or also known as artificial intelligence is one branch of computer science to provide a knowledge of computers in order to be able to complete tasks or think like humans. One example of artificial intelligence that can be applied to games is Path Finding. Path Finding is one of the artificial intelligence that is used to determine the shortest blend between the starting point and end point. Fuzzy logic is the study of uncertainty. Fuzzy logic is also able to map an input space into an output space precisely. The method used in this research is the prototype meode where the steps taken are analyzing the needs, designing the prototype, implementing, and testing. The main objective to be achieved from research is to compare the performance of the Dijkstra algorithm and the A Star algorithm for the completion of the Pac-Man game. The results obtained for the Djikstra algorithm are 2 times failed, and 1 time succeeded in completing the game with a score of 4100, 3350, 3940, while for the A Star algorithm it got results 2 times successfully, and 1 time failed with a score of 4300, 2350, 3450. From The second algorithm used to solve the PAC-MAN game by getting the best score is the A Star algorithm.

Keywords: Path Finding, Dijkstra, A Star, fuzzy logic, Artificial Inteligence

1. PENDAHULUAN

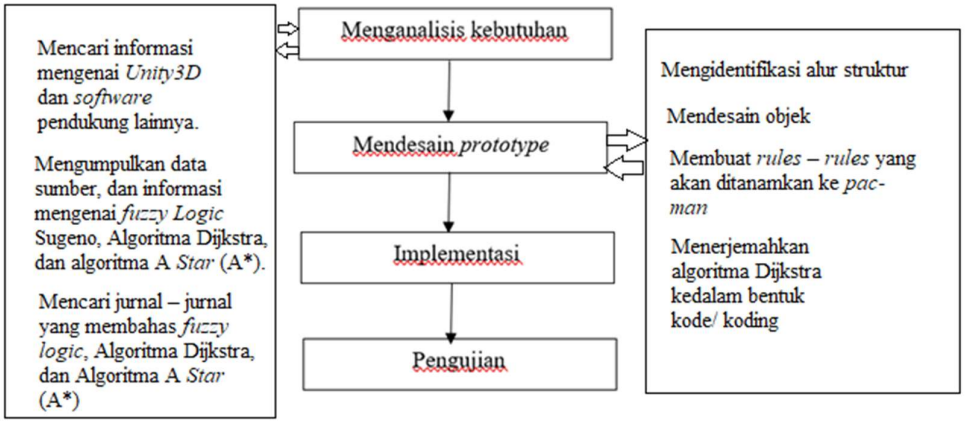
Dalam suatu permainan terdapat sebuah logika yang bertujuan memberikan alur dari jalannnya permainan. Logika yang mengatur alurnya digunakan sebuah algoritma. Algoritma tersebut menggambarkan akan alur yang digunakan, Algoritma yang digunakan dalam penelitian ini adalah Algoritma *Dijkstra* dan Algoritma *A Star.* Untuk logika yang digunakan dalam alur tersebut adalah *fuzzy* yang membantu mengambil keputusan untuk memecahkan masalah. *Path finding* merupakan sebuah pencarian jalur yang biasanya digunakan untuk menentukan rute yang akan dilalui, path finding menggunakan algoritma untuk melakukan pengambilan rute, salah satu contoh dati algoritma path finding adalah *Dijkstra*, dan *A Star*. Pada penelitian ini *digunakan path finding*, dan logika *fuzzy* yang akan diterapkan pada karakter utama *Pac-Man,* sehingga permainan *Pac-Man* dapat diselesaikan.

2. METODE PENELITIAN

Pada penelitian ini akan menggunakan metode penelitian *Prototype* agar proses pembuatan ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan pada tahap awal.

1. Metode *Prototype*

prototyping merupakan metode pengembangan perangat lunak, yang berupa model fisik kerja sistem dan berfungsi sebagai versi awal dari sistem, Agar proses pembuatan prototype ini berhasil dengan baik adalah dengan mendefinisikan aturan-aturan pada tahap awal, *Prototype* akan dihilangkan atau bahkan pada bagiannya sehingga sesuai dengan perencanaan dan analisis yang dilakukan oleh pengembang sampai dengan ujicoba dilakukan secara simultan seiiring dengan proses pengembangan[1].



Gambar 1 Tahapan – Tahapan Metodologi penelitian

Tahapan-tahapan yang akan dilakukan adalah sebagai berikut :

a) Menganalisis kebutuhan
   Pada tahap ini, *platform* ditentukan dan dilakukan identifikasi kebutuhan sistem yang akan dibuat meliputi tujuan, manfaat, adapun tahapan yang dilakukan:
   1. Mencari informasi mengenai Unity3D dan *software* pendukung lainnya. Referensinya dapat diperoleh dari buku, jurnal, maupun situs-situs di internet.
   2. Mengumpulkan data sumber, dan informasi mengenai *fuzzy Logic* Sugeno, Algoritma *Dijkstra*, dan algoritma A *Star* (A*).

3. Mencari jurnal – jurnal yang membahas *fuzzy logic*, Algoritma *Dijkstra*, dan Algoritma A *Star* (A*).

b) Mendesain *prototype*
Pada tahap ini dilakukan pembangunan *prototyping* dengan membuat peracangan sementara.
Dengan tahapan yang dilakukan :
1. Mengidentifikasi alur struktur yang berjalan dalam pembuatan *game*.
2. Mendesain objek yang akan dibuat nantinya, adapun objek yang akan dibuat berupa pac-man/pemain, Ghost, Pellet/ titik – titik kuning, dan pembuatan *map* dalam bentuk *grid – gird* kotak.

Gambar 2 Pac- man dan *Ghost*

Gambar 3 *Pellet*

3. Membuat *rules – rules* yang akan ditanamkan ke pac-man.
Rules-rules yang akan dibuat menggunakan 4 *variable* sebagai penentu :
   1. *Pellet Distance : Near, Medium, Far*
   2. *Ghost distance : Near, Medium, Far*
   3. *Health : Low, Medium, High*
   4. *Power duration: Short, Medium, Long*
4. Menerjemahkan algoritma *Dijkstra* dan algoritma A *Star* kedalam bentuk kode.

c) Implementasi
Pada tahap ini, prototyping yang sudah dirancang sebelumnya akan diimplementasikan agar menjadi aplikasi yang sudah sesuai dengan rancangan, dengan tahapan yang dilakukan :
1. Membangun/ membuat game menggunakan objek – objek yang sudah dibuat.

Gambar 4 Hasil Bangun Game

2. Penerapan algoritma *Dijkstra* dan algoritma A *Star* (A*) untuk pencarian *rute* terpendek ke dalam pac-man. Algoritma tersebut dapat dilihat pada gambar 5 berikut ini :

```
function Dijkstra(Graph, source):
      for each vertex v in Graph:
// Initializations
          dist[v] := infinity ;
// Unknown distance function from

// source to v
          previous[v] := undefined ;
// Previous node in optimal path
      end for
// from source

      dist[source] := 0 ;
// Distance from source to source
      Q := the set of all nodes in
Graph ;                         // All
nodes in the graph are

// unoptimized - thus are in Q
      while Q is not empty:
// The main loop
          u := vertex in Q with
smallest distance in dist[] ;    //
Start node in first case
          remove u from Q ;
          if dist[u] = infinity:
              break ;
// all remaining vertices are
          end if
// inaccessible from source

          for each neighbor v of u:
// where v has not yet been

// removed from Q.
              alt := dist[u] +
dist_between(u, v) ;
              if alt < dist[v]:
// Relax (u,v,a)
                  dist[v] := alt ;
                  previous[v] := u ;
                  decrease-key v in
Q;                              //
Reorder v in the Queue
              end if
          end for
      end while
  return dist;
```

```
function reconstruct_path(cameFrom, current)
    total_path := {current}
    while current in cameFrom.Keys:
        current := cameFrom[current]
        total_path.prepend(current)
    return total_path

// A* finds a path from start to goal.
// h is the heuristic function. h(n) estimates
the cost to reach goal from node n.
function A_Star(start, goal, h)
    // The set of discovered nodes that may need
to be (re-)expanded.
    // Initially, only the start node is known.
    // This is usually implemented as a min-heap
or priority queue rather than a hash-set.
    openSet := {start}

    // For node n, cameFrom[n] is the node
immediately preceding it on the cheapest path
from start
    // to n currently known.
    cameFrom := an empty map

    // For node n, gScore[n] is the cost of the
cheapest path from start to n currently known.
    gScore := map with default value of Infinity
    gScore[start] := 0

    // For node n, fScore[n] := gScore[n] +
h(n). fScore[n] represents our current best
guess as to
    // how short a path from start to finish can
be if it goes through n.
    fScore := map with default value of Infinity
    fScore[start] := h(start)

    while openSet is not empty
        // This operation can occur in O(1) time
if openSet is a min-heap or a priority queue
        current := the node in openSet having
the lowest fScore[] value
        if current = goal
            return reconstruct_path(cameFrom,
current)

        openSet.Remove(current)
        for each neighbor of current
            // d(current,neighbor) is the weight
of the edge from current to neighbor
            // tentative_gScore is the distance
from start to the neighbor through current
            tentative_gScore := gScore[current]
+ d(current, neighbor)
            if tentative_gScore <
gScore[neighbor]
                // This path to neighbor is
better than any previous one. Record it!
                cameFrom[neighbor] := current
                gScore[neighbor] :=
tentative_gScore
                fScore[neighbor] :=
gScore[neighbor] + h(neighbor)
                if neighbor not in openSet
                    openSet.add(neighbor)

    // Open set is empty but goal was never
reached
    return failure
```

Gambar 5. Algoritma *Dijkstra* dan *A Star*

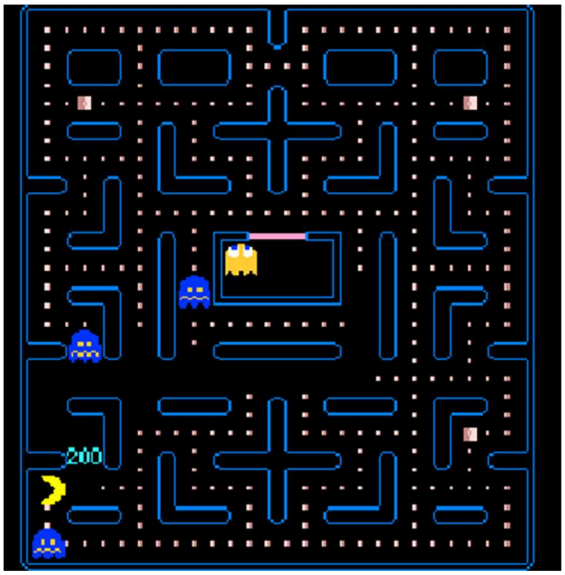
Jurnal Algoritme
Vol. 1, No. 1, Oktober 2020, Hal. 12 – 20

3. Penerapan *rules – rules* logika *fuzzy* yang sudah dibuat ke dalam pac-man.

d) Pengujian
Tahap pengujian merupakan tahapan untuk melakukan pengujian penerapan algoritma *Dijkstra*, algoritma A *Star* (A*) dan logika *fuzzy* Sugeno. Pada algoritma Dijkstra pengujian dilakukan dengan teknik pengujian *black-box Performance Testing* untuk menguji fungsionalitas aplikasi apakah sesuai dengan hasil yang diharapkan, dengan parameter pengujian berupa *score* akhir dari permaian. Tujuan dari metode *Black Box Testing* ini adalah untuk menemukan kesalahan fungsi pada program.Pengujian dengan metode *Black Box Testing* dilakukan dengan cara memberikan sejumlah input pada program. Input tersebut kemudian diproses sesuai dengan kebutuhan fungsionalnya untuk melihat apakah program aplikasi dapat menghasilkan output yang sesuai dengan yang diinginkan dan sesuai pula dengan fungsi dasar dari program tersebut.

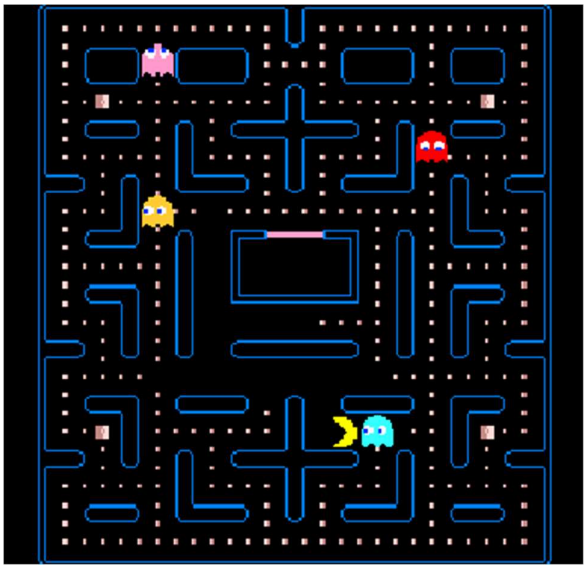
3. HASIL DAN PEMBAHASAN

1. Implementasi Algoritma
Pada tahap ini akan dilakukan proses implementasi algoritma *Dijkstra* dan A *Star* ke dalam 3 perilaku pac-man sebagai berikut :
1. Kejar : Kondisi pacman mengejar *Ghost*.



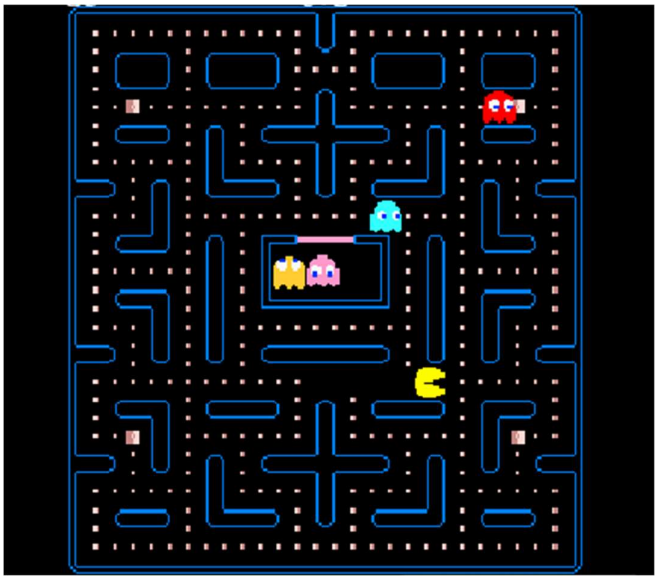
Gambar 6. Perilaku Kejar

2. Kabur : Kondisi pac-man menjauhi *Ghost*.



Gambar 7 Perilaku Kabur

3. Makan : Kondisi di mana pacman mentargetkan *pellet*.



Gambar 8 Perilaku Makan

Hasil Pengujian yang diperoleh dilakukan sebanyak 3 kali.

Tabel 1 Kondisi

No	Prilaku	*Start*	*Goal/ Target*
1	Kejar	Pacman	*Ghost*
2	Kabur	Pacman	*Ghost*
3	Makan	Pacman	*Ghost*

2. Pengujian Dan Hasil

Pada tahap pengujian dilakukan dengan menggunakan metode *Black Box,* hasil pengujian berupa tingkat kesusuaian pengujian dengan hasil yang di harapkan.

Berikut ini adalah tabel pengujian untuk algoritma *Dijkstra* dan algoritma *A Star*

Tabel 2 Hasil Pengujian Algoritma

No	Test Case	Hasil yang Diharapkan	Hasil yang Didapatkan	Dijkstra	A Star	Kesimpulan
1	Mengejar *Pellet* dan *Power pellet*	Pacman dapat mengejar *Pellet*	Pacman mengejar *Pellet*	✓	✓	Diterima
2	Kabur dari *Ghost*	Pacman dapat Menjauhi *Ghost*	Pacman menjauhi *Ghost*	✓	✓	Diterima
3	Memakan *ghost*	Pacman dapat memakan *Ghost*	Pacman memakan *Ghost* saat *Power Pellet* sedang berlangsung	✓	✓	Diterima

Pada tahap pengujian dilakukan dengan melakukan 3 kali percobaan. Untuk data data yang diuji berupa waktu, *score*, dan sisa *health*.

Tabel 3 Hasil Pengujian Algoritma *Djikstra*

NO	*Score*	Waktu	*Health*	*Status*
1	4100	1 menit 11 detik	0	Gagal dengan sisa *Health* 0 dan sisa *pellet 2*
2	3350	1 menit 12 detik	0	Gagal dengan sisa *Health* 0 dan sisa *pellet 35*
3	3940	1 menit 4 detik	3	Dapat Menyelesaikan Permainan

Dari tabel di atas didpat 2 kali gagal dan 1 kali berhasil dengan waktu rata-rata yang ditempuh 1 menit, dan *score* 4100, 3350, 3940. Untuk percobaaan pertama mendapat kegagalan dengan dengan sisa *health* 0 dan sisa *pellet 2, dan untuk percobban kedua* mendapat kegagalan

dengan dengan sisa *health* 0 dan sisa *pellet 35,* sedangkan untuk percobaan ketiga mendapat keberhasilan dengan sisa *Health* 3 dan sisa *Pellet 0.*

Tabel 4 Hasil Pengujian Algoritma *A Star*

NO	*Score*	Waktu	*Health*	*Status*
1	4300	1 menit 15 detik	2	Dapat Menyelesaikan Permainan
2	2350	52detik	0	Gagal dengan sisa *Health* 0 dan sisa *pellet 60*
3	3450	1 menit 30 detik	1	Dapat Menyelesaikan Permainan

Dari tabel di atas didpat 2 berhasil dan 1 kali gagal dengan waktu rata-rata yang ditempuh 1 menit, dan *score* 4300, 2350, 3450. Untuk percobaaan pertama mendapat keberhasilan dengan dengan sisa *health* 3 dan sisa *pellet 0, dan untuk percobban kedua* mendapat kegagalan dengan dengan sisa *health* 0 dan sisa *pellet 60,* sedangkan untuk percobaan ketiga mendapat keberhasilan dengan sisa *Health* 1 dan sisa *Pellet 0.*

KESIMPULAN

Berdasarkan hasil pengujian dapat disimpulkan bahwa :

1. Dari hasil perbadingan yang dilakukan algoritma A Star mendapatkan hasil *score*, waktu, dan *health* yang tinggi.
2. Algoritma A Star dapat menyelesaikan permainan sebanyak 2 kali dan 1 kali mengalami kegagalan, dan Algoritma Djikstra dapat menyelesaikan permainan sebanyak 1 kali dan 2 kali mengalami kegagalan.
3. Dari kedua Algoritma yang digunakan untuk menyelesaikan permaian *Pac-Man* dengan mendapatkan *score* terbaik adalah algoritma *A Star* yakni sebesar 4300.

5. SARAN

Berdasarkan kesimpulan tersebut, untuk menghasilkan hasil yang lebih baik, berikut saran yang nantinya dapat bermanfaat untuk penelitian selanjutnya.

1. Sebaiknya menggunakan algortma A Star untuk melakukan *path finding* pada maze.
2. *Sample* untuk percobaan bisa diperbanyak untuk mendapatkan hasil yang lebih baik lagi.
3. Hasil penelitian ini dapat digunakan untuk melakukan perbandingan AI vs AI.

UCAPAN TERIMA KASIH

Penulis juga banyak mengucapkan terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung, juga kepada pihak-pihak yang telah memberikan bimbingan, pengarahan, maupun ide-ide untuk penulis selama proses penyelesaian penelitian ini.

DAFTAR PUSTAKA

[1] Ogedebe, P.M.,& Jacob, B.P. , 2012, Software Prototyping: A Strategy to Use When User Lacks Data Processing Experience, *ARPN Journal of Systems and Software*. VOL. 2, NO. 6, 2012, http://scientificjournals.org/journalofsystemsandsoftware/archive/vol2no6/vol2no6_4.pdf

[2] Fakhri. , 2008, Penerapan Algoritma Dijkstra Dalam Pencarian Solusi Maximum Flow Problem. *Makalah IF2251 Strategi Algoritmik*

[3] Barnouti, N.H., Al-Dabbagh, S.S.M. and Naser, M.A.S., 2016, Pathfinding in Strategy Games and Maze Solving Using A* Search Algorithm.*Journal of Computer and Communications*, 4, 15-25.

[4] Rahakbauw, D, L. , 2015, Penerapan Logika Fuzzy Metode Sugeno Untuk Menentukan Junlah Produksi Roti Berdasarkan Data Persediaan Dan Jumlah Permintaan (Studi Kasus : Prabik Roti Sarinda Ambon), *Jurnal Ilmu Matematika dan Terapan*, Volume. 9 Nomor. 2, Desember 2015, Hal. 121 – 134.

[5] Muhammad, Wali., Suhartono, Vincent., Soeleman, Arief, Muhammad. , 2014, Penentuan Jalur Pergerakan Dan Perilaku Perang Karakter Bukan Pemain Menggunakan Algoritma A* Dan Metode Logika Fuzzy Pada Game Simulator Tank, *Jurnal Teknologi Informasi*, Volume. 10 Nomor. 2, Oktober 2014.

[6] Eka, F, A., Wiwien, Hadikurniawati. , 2018, *Implementasi Algoritma Dijkstra Untuk Mencari Rute Terpendek Antar Kantor Dan Estimasi Penggunaan Bahan Bakar Kendaraan (Studi Kasus Pt. Telkom Indonesia Regional Iv Jateng-Diy)*, Universitas Seminar Nasional Multi Disiplin Ilmu Dan Call For Papers, Stikubank, Semarang, 25 Juli 2018.

[7] Syamsuddin, M, Y., Hanifah, Muslimah, Az-Zahra., Diah, Harnoni, Apriyanti. , 2017, Implementasi Algoritma Dijkstra Dalam Menemukan Jarak Terdekat dari Lokasi Pengguna Ke Tanaman Yang Di Tuju Berbasis Android (Studi Kasus di Kebun Raya Purwodadi), *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, Vol. 1, No. 12, Desember 2017.

[8] Fathurochman, Dery, Witanti, Wina dan Yuniarti, Rezki, 2014, Seminar Nasional Informatika 2014, *Perancangan Game Turn Based Strategy Menggunakan Logika Fuzzy Dan Naive Bayes Classifier*, UPN "Veteran" Yogyakarta, 12 Agustus 2014.

[9] Marlina, Leni., Amin, Suyitno., Mashuri. (2017). Penerapan Algoritma Dijkstra Dan Floyd-Warshall Untuk Menentukan Rute Terpendek Tempat Wisata Di Batang, *Unnes Journal of Mathematics*, Vol. 6, No. 1, Mei 2017.