Aplikasi Algoritma Slime Mould dalam Optimasi Graf

Muhamad Rafli Rasyiidin - 13522088[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522088@std.stei.itb.ac.id

***Abstract*—Algoritma Slime Mould merupakan salah satu metode yang dapat digunakan untuk menyelesaikan permasalahan optimasi. Algoritma ini terinspirasi dari perilaku slime mould untuk menemukan rute terpendek dalam mencari makanan. Algoritma ini biasanya digunakan dalam optimasi rute perjalanan, desain jaringan, dan pengoptimasian graf. Makalah ini membahas salah satu penggunaan algortima slime mould yaitu pengoptimasian graf.**

***Keywords*—Graf, Slime Mould, Optimasi, Physarium**

I. PENDAHULUAN

Dalam era perkembangan teknologi yang pesat, optimasi graf menjadi salah satu bahasan yang cukup penting untuk menghadapi berbagai permasalahan dalam kehidupan. Ada beberapa metode yang dapat digunakan untuk mengoptimasi suatu graf. Salah satu metode tersebut adalah algoritma *slime mould*. Algoritma ini terinspirasi dari perilaku organisme jamur lendir atau *slime mould*.

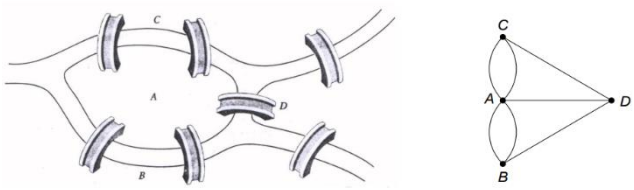
Slime Mould atau *Physarium polycephalum* merupakan organisme yang memiliki kemampuan adaptasi dan penyebaran yang luar biasa. *Slime mould* dapat mendeteksi sumber makanan yang ada di sekitarnya dan membuat jalur ke arah sumber makanan tersebut. Kemampuan ini menjadikan *slime mould* sebagai sumber inspirasi bagi pengembangan algoritma optimasi yang efisien. Para peneliti memanfaatkan *slime mould* untuk menyelesaikan permasalahan kompleks, seperti pencarian rute optimal, perencanaan jaringan, optimasi graf, dan permasalahan optimasi lainnya.

Pada makalah ini akan dilakukan analisis dan pembahasan mengenai penggunaan algoritma *slime mould* dalam optimasi graf, serta simulasinya.

II. LANDASAN TEORI

A. Graf

Graf merupakan suatu struktur yang menghubungkan satu simpul (*node*) dengan simpul lainnya melalui sebuah sisi (*vertices*). Graf biasanya digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut. Graf pertama kali diperkenalkan pada tahun 1736 oleh seorang matematikawan Swiss Bernama Leonhard Euler untuk menyelesaikan permasalahan jembatan Königsberg. Permasalahan jembatan Königsberg adalah permasalahan dalam menentukan kemungkinan melalui tujuh jembatan tepat sekali dan kembali ke tempat semula.

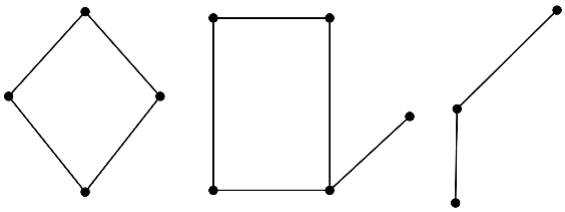


Gambar 2.1. Permasalahan jembatan Königsberg dan graf yang merepresentasikannya
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/)

Secara matematis, graf dapat didefinisikan sebagai berikut:
$G = (V, E)$
V merupakan himpunan tidak kosong dari simpul-simpul (*vertices*). $V = \{v_1, v_2, ..., v_n\}$.
E merupakan himpunan sisi (*edges*) yang menghubungkan sepasang simpul. $E = \{e_1, e_2, ..., e_n\}$.

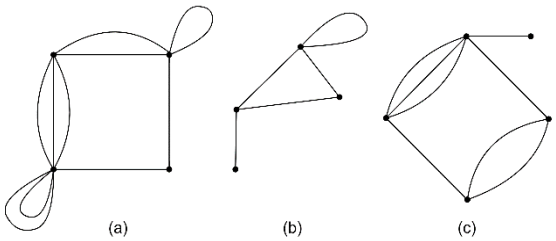
Graf digolongkan menjadi dua jenis berdasarkan ada tidaknya gelang atau sisi ganda:

1. Graf sederhana (*simple graph*)
   Graf sederhana adalah graf yang tidak mengandung gelang maupun sisi ganda.



Gambar 2.2. Graf sederhana
(Sumber: dokumen penulis)

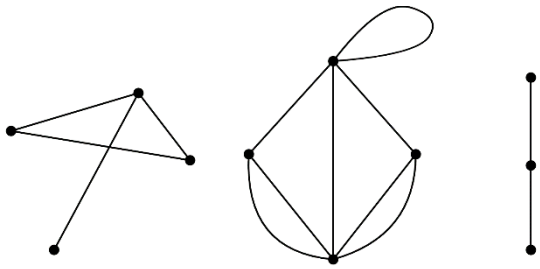
2. Graf tak-sederhana (*unsimple graph*)
   Graf tak-sederhana adalah graf yang mengandung gelang atau sisi ganda. Graf yang mengandung gelang dinamakan graf semu (*pseudograph*), sedangkan graf yang mengandung sisi ganda dinamakan graf ganda (*multigraph*).



Gambar 2.3. (a) Graf tak-sederhana (b) Graf semu (c) Graf ganda
(Sumber: dokumen penulis)

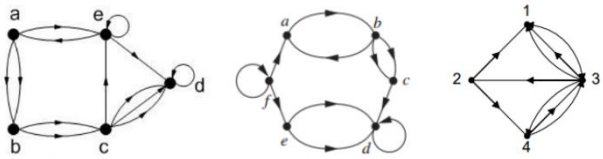
Berdasarkan orientasi arah pada sisi graf, graf dibagi menjadi dua jenis:

1. Graf tak-berarah (*undirected graph*)
   Graf tak-berarah adalah graf yang sisinya tidak mempunyai orientasi arah.



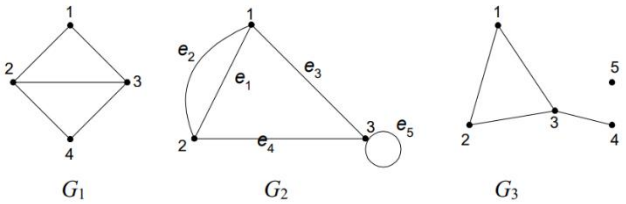
Gambar 2.4. Graf tak-berarah
(Sumber: dokumen penulis)

2. Graf berarah (*directed graph*)
   Graf berarah adalah graf yang setiap sisinya memiliki orientasi arah.



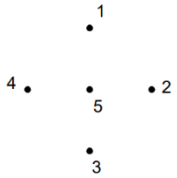
Gambar 2.5. Graf berarah
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/)

B. Terminologi Graf



Gambar 2.6. Terminologi graf
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/)

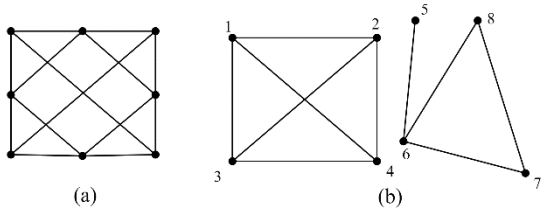
1. Ketetanggaan (*Adjacent*)
   Dua buah simpul dikatakan bertetangga jika keduanya terhubung langsung oleh suatu sisi. Contohnya terdapat pada graf G1 gambar 2.6. Pada graf G1, simpul 1 bertetangga dengan simpul 2 dan 3.
2. Bersisian (*Incidency*)
   Suatu sisi $e = (v_j, v_k)$ dikatakan bersisian jika sisi tersebut menghubungkan simpul v_j, dengan v_k.. Sebagai contoh, tinjau graf G2 pada gambar 2.6. Pada graf G2, e_3 bersisian dengan simpul 1 dan 3.
3. Simpul Terpencil (*Isolated Vertex*)
   Simpul terpencil adalah suatu simpul yang tidak memiliki sisi yang bersisian dengannya. Sebagai contoh, tinjau graf G3 pada gambar 2.6. Pada graf G3, simpul 5 merupakan simpul terpencil.
4. Graf Kosong (*Null Graph*)
   Graf kosong merupakan graf yang himpunan sisinya berupa himpunan kosong (tidak memiliki sisi, hanya memiliki simpul).



Gambar 2.7. Graf kosong
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/)

5. Derajat (*Degree*)
   Derajat adalah jumlah sisi yang bersisian dengan suatu simpul. Derajat suatu simpul dinotasikan dengan *d(v)*. Tinjau graf G1 pada gambar 2.6. Pada graf tersebut, simpul 1 dan 4 memiliki derajat 2, sedangkan simpul 2 dan 3 memiliki derajat 3.
6. Lintasan (*Path*)
   Lintasan adalah barisan selang-seling antara simpul dan sisi sedemikian sehingga menghubungkan simpul awal v_0 dengan simpul akhir v_n. Tinjau graf G_2 pada gambar 2.6. Lintasan 1, 2, 3 merupakan lintasan dari simpul 1 ke 3 yang melalui sisi e_1 dan e_4 atau e_2 dan e_4.
7. Siklus (*Cycle*) atau Sirkuit (*Circuit*)
   Siklus atau sirkuit adalah sebuah lintasan yang berawal dan berakhir pada simpul yang sama. Tinjau graf G1 pada gambar 2.6. Lintasan 1, 2, 3, 4 adalah sebuah siklus.
8. Keterhubungan (*Connected*)
   Dua buah simpul v_i dan simpul v_2 disebut terhubung jika terdapat lintasan dari v_1 ke v_2. Suatu graf disebut terhubung jika terdapat lintasan dari v_i ke v_j pada setiap pasang simpul v_i dan v_j dalam

himpunan V.

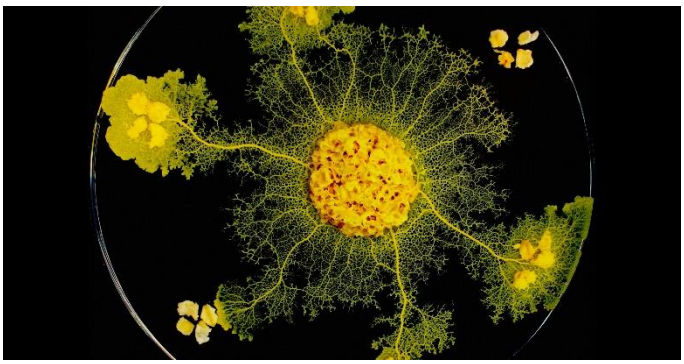


Gambar 2.8. (a) Graf terhubung (b) Graf tak-terhubung
(Sumber: dokumen penulis)

9. Upagraf (*Subgraph*) dan Komplemen Upagraf
   Sebuah graf $G_1 = (V_1, E_1)$ disebut upagraf dari $G = (V, E)$ jika $V_1 \subseteq V$ dan $E_1 \subseteq E$. Upagraf G_1 dikatakan upagraf merentang jika ia mengandung semua simpul pada graf *G*. Sebagai contoh, tinjau graf (a) dan graf (b) yang berbentuk kotak pada gambar 2.8. Graf (b) yang berbentuk kotak tersebut merupakan upagraf dari graf (a).
10. *Cut-Set*
    Cut-Set adalah himpunan sisi dari graf terhubung *G* yang jika dibuang akan menyebabkan *G* menjadi tidak terhubung. Tinjau graf G_3 pada gambar 2.6. Sisi *e* = (1, 3) merupakan *cut-set* dari graf G_3.
11. Graf Berbobot
    Graf berbobot merupakan graf yang setiap sisinya memiliki sebuah harga (bobot).

C. Algoritma *Slime Mould*

Algoritma *Slime Mould* atau algoritma *physarium* adalah suatu algoritma yang terinspirasi dari perilaku jamur lendir (*slime mould*) dalam mencari makanan. Algoritma ini dapat dimanfaatkan untuk menyelesaikan masalah optimasi atau perencanaan jalur.



Gambar 2.9. *Slime Mould*
(Sumber: https://www.quantamagazine.org/slime-molds-remember-but-do-they-learn-20180709/)

Dari perilaku jamur tersebut, dibuatlah sebuah model matematika:

1. *Approach Food*
   Slime mould dapat merasakan dan mendekati sumber makanan berdasarkan baunya yang tersebar di udara. Secara matematis, perilaku ini dimodelkan sebagai berikut:

$$X(t+1) = \begin{cases} X_b(t) + vb \cdot \left(W \cdot X_A(t) - X_B(t)\right), & r \le p \\ vc \cdot X(t), & r \ge p \end{cases} \quad (1)$$

(Sumber: https://www.baeldung.com/cs/slime-mould-algorithm)

Pada persamaan (1), *vb* adalah parameter yang memiliki rentang nilai dari -1 hingga 1 dan *vc* adalah parameter yang memiliki rentang nilai dari 0 hingga 1. *W* adalah adalah bobot atau diameter dari jaringan pembuluh *slime mould*, *t* adalah iterasi saat ini, X_A dan X_B adalah dua lokasi yang dipilih secara acak dari *slime mould*, X adalah lokasi dari *slime mould*, dan X_b adalah lokasi sumber makanan dengan konsentrasi bau paling tinggi saat ini. Selain itu, terdapat persamaan untuk *p*, *vb*, dan *W* yang didefinisikan sebagai berikut:

$$p = \tanh\,|S(i) - DF| \quad (2)$$

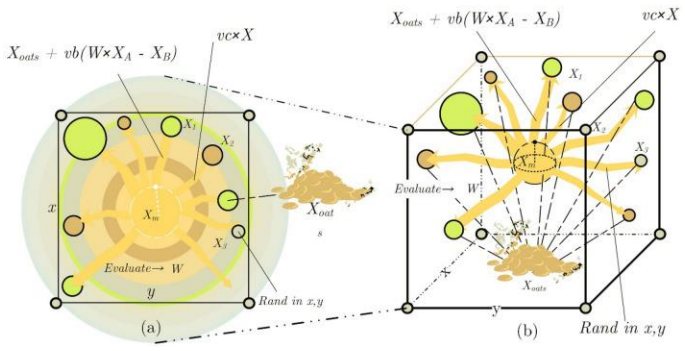
$$vb = [-a \cdot a] \quad (3)$$

$$a = \operatorname{arctanh}\left(-\left(\frac{t}{max_t}\right) + 1\right) \quad (4)$$

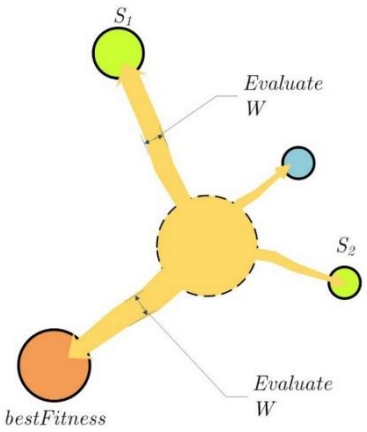
$$W(SmellIndex(i)) = \begin{cases} 1 + r \cdot \log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & condition \\ 1 - r \cdot \log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & others \end{cases} \quad (5)$$

(Sumber: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9838547/)

Pada persamaan (2), *DF* merepresentasikan kualitas sumber makanan terbaik yang dicapai dari seluruh iterasi dan *S(i)* menunjukkan kualitas sumber makanan dari lokasi *X*. Pada persamaan (5), W merupakan bobot yang membantu *slime moulds* untuk mempercepat gerakan ke arah sumber makanan berkualitas tinggi dan memperlambat pergerakannya ketika konsentrasi makanan rendah. Selain itu, pada persamaan (5), *SmallIndex* merepresentasikan kualitas sumber makanan yang disusun terurut membesar, *F* adalah nilai kualitas sumber makanan terburuk yang diperoleh saat iterasi, *bF* menunjukkan kualitas sumber makanan optimal yang diperoleh saat iterasi, *r* menandakan nilai acak pada interval [0..1], max_t menunjukkan jumlah iterasi maksimum, dan *condition* menunjukkan persamaan yang digunakan ketika *S(i)* menempati peringkat separuh teratas dari populasi.



Gambar 2.10. Cara kerja *slime mould*
(Sumber: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9838547/)



Gambar 2.11. Pengukuran kesuburan
(Sumber: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9838547/)

1. *Wrap Food*
   Berdasarkan definisi-definisi pada bagian 1, diperoleh persamaan matematis untuk memperbarui posisi *slime moulds*:

$$X^* = \begin{cases} rand \cdot (UB - LB) + LB, & rand \le z \\ X_b(t) + vb \cdot \left(W \cdot X_A t - X_B(t)\right), & r < p \\ vc \cdot X(t), & r \ge p \end{cases}$$

III. Analisis dan Pembahasan

A. Penerapan Algoritma *Slime Mould*

Pada bagian ini, akan dijelaskan cara kerja dari algoritma *slime mould* yang telah dibahas pada bagian landasan teori.

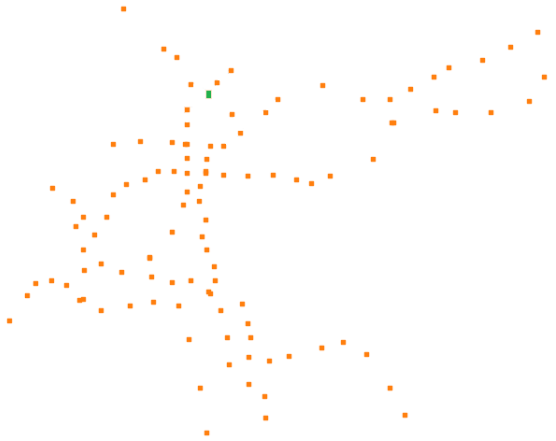
```
Pseudocode Algoritma Slime Mould

Input(MaxIterate, PopulationSize, z)
Initialize Slime Mould Position {Xi|i =
1,2,…PopulationSize}
t <- 0
while t < MaxIterate do
  CalculateFitness;
  UpdateBestFitness Xb;
  Calculate W;
  AgentIndex <- 0
  while AgentIndex < PopulationSize do
    Update p, vb, vc;
    Update Mould Positions;
    AgentIndex <- AgentIndex + 1;
  t <- t + 1
```

(Sumber: https://www.baeldung.com/cs/slime-mould-algorithm)

Pseudocode di atas merupakan gambaran dari algoritma *slime mould* jika dibuat menjadi program. Ketika program dimulai, program akan menerima masukan jumlah maksimal iterasi yang ingin dilakukan (jumlah iterasi sama dengan jumlah langkah yang akan dilakukan oleh *slime mould*), jumlah sumber makanan, dan z. Selanjutnya, lakukan inisialisasi posisi awal *slime mould* (X) dan lakukan iterasi sebanyak MaxIterate. Dalam setiap iterasi tersebut, program akan mencari kualitas sumber makanan terbaik (X_b) dan menyimpannya dalam suatu variable, misalnya Xb. Kemudian, program akan menghitung bobot (W) dari kualitas sumber makanan menggunakan persamaan (5). Terakhir, program akan memperbarui posisi *slime mould* terhadap setiap sumber makanan yang ada.

B. Simulasi

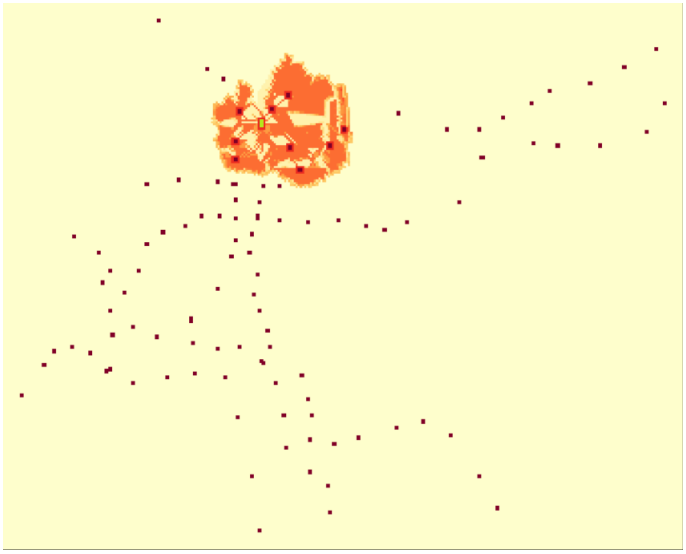
Dari *pseudocode* di atas, dapat dibuat sebuah simulasi yang menggambarkan cara kerja *slime mould*. Dalam simulasi ini, akan digunakan graf kosong sebagai berikut:



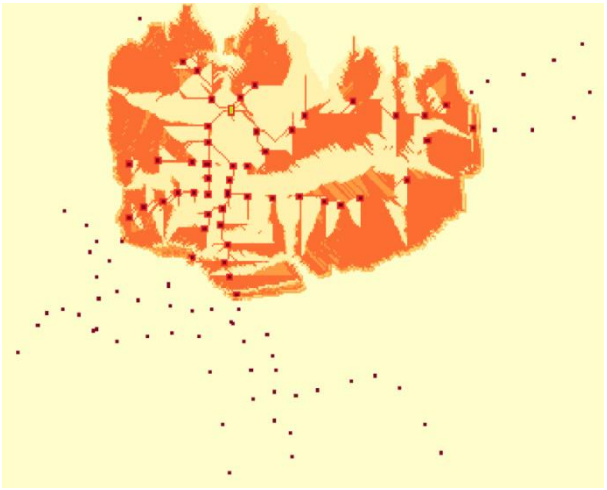
Gambar 3.1. Graf kosong untuk simulasi
(Sumber: https://github.com/MoeBuTa/SlimeMould)

Simpul berwarna hijau melambangkan posisi awal dari *slime mould* dan simpul lainnya menggambarkan sumber makanan yang akan dituju oleh *slime mould*. Ketika *slime mould* mencapai salah satu simpul, simpul tersebut akan berubah menjadi posisi terbaru *slime mould*. Selanjutnya, ketika program dijalankan, program akan membentuk beberapa sisi yang menghubungkan simpul-simpul tersebut. Sisi-sisi tersebut menggambarkan jalur terdekat dari posisi *slime mould* ke sumber makanan yang dituju.

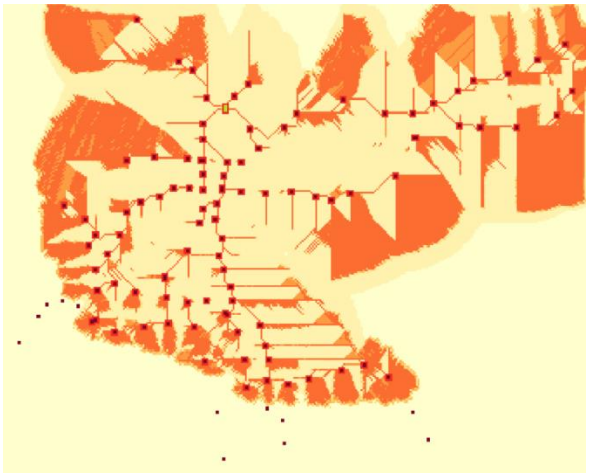
Berikut ini merupakan beberapa simulasi algoritma *slime mould* dengan jumlah iterasi sebanyak 50, 150, 250, dan 350.



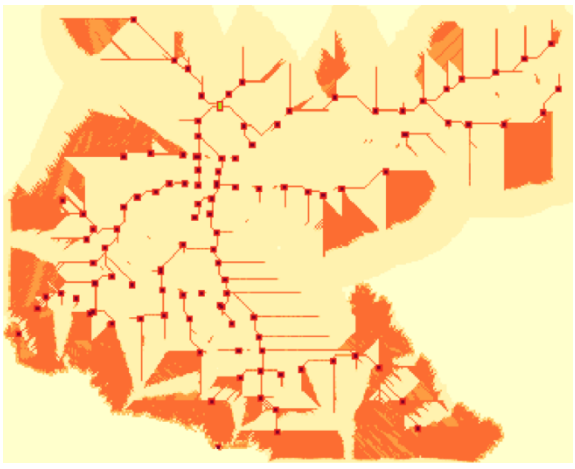
Gambar 3.2. Simulasi *slime mould* dengan 50 langkah atau 50 iterasi
(Sumber: https://github.com/MoeBuTa/SlimeMould)



Gambar 3.3. Simulasi *slime mould* dengan 150 langkah atau 150 iterasi
(Sumber: https://github.com/MoeBuTa/SlimeMould)



Gambar 3.4. Simulasi *slime mould* dengan 250 langkah atau 250 iterasi
(Sumber: https://github.com/MoeBuTa/SlimeMould)



Gambar 3.5. Simulasi *slime mould* dengan 350 langkah atau 350 iterasi
(Sumber: https://github.com/MoeBuTa/SlimeMould)

Dari simulasi tersebut, didapatkan graf yang setiap simpulnya telah terhubung dengan iterasi sebanyak 350 kali. Berikut merupakan gambar graf yang didapat dari iterasi sebanyak 350 kali:

Gambar 3.5. Graf yang didapatkan dari simulasi algoritma *slime mould*
(Sumber: https://github.com/MoeBuTa/SlimeMould)

Pada simulasi tersebut, algoritma *slime mould* menghubungkan setiap simpul yang ada berdasarkan persamaan-persamaan yang telah disebutkan pada landasan teori. Untuk menghubungkan seluruh simpul, diperlukan jumlah iterasi tertentu. Jika jumlah iterasi tidak cukup, graf hanya akan memiliki beberapa simpul yang terhubung atau bahkan tidak ada simpul yang terhubung sama sekali (*slime mould* belum mencapai sumber makanan).

IV. KESIMPULAN

Dari analisis dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa algoritma *slime mould* dapat digunakan untuk mengoptimasi graf. Graf yang akan dioptimasi dapat diubah menjadi graf kosong. Kemudian, graf tersebut akan diproses oleh algoritma *slime mould* dan menghasilkan graf yang lebih optimal. Namun, untuk menghasilkan graf yang optimal, diperlukan jumlah iterasi tertentu agar seluruh simpul dapat terhubung. Oleh karena itu, salah satu kekurangan dari algoritma ini adalah jumlah iterasi yang diperlukan tidak diketahui sehingga akan sedikit sulit untuk membentuk graf yang optimal dengan jumlah iterasi yang minimal.

V. PENUTUP

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan makalah ini dengan lancar. Penulis juga mengucapkan terima kasih kepada Ibu Fariska Zakhralativa Ruskanda, S.T., M.T., atas bimbingannya selama perkuliahan sehingga penulis dapat menyelesaikan perkuliahan dan makalah ini dengan baik. Selain itu, penulis juga ingin mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir atas materi dan referensi yang telah disediakan pada situs beliau. Terakhir, penulis ingin mengucapkan terima kasih kepada orang tua, keluarga, teman, dan seluruh pihak yang terlibat dalam

penulisan makalah ini. Penulis berharap makalah ini dapat memberikan manfaat bagi para pembaca.

REFERENCES

[1] https://www.quantamagazine.org/slime-molds-remember-but-do-they-learn-20180709/ [diakses pada 9 Desember 2023]

[2] Gharehchopogh, F. S., Ucan, A., Ibrikci, T., Arasteh, B., & Isik, G. (2023). Slime Mould Algorithm: A Comprehensive Survey of Its Variants and Applications. *Archives of computational methods in engineering : state of the art reviews*, *30*(4), 2683–2723. https://doi.org/10.1007/s11831-023-09883-3H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[3] https://github.com/MoeBuTa/SlimeMould diakses pada 8 Desember 2023

[4] https://www.baeldung.com/cs/slime-mould-algorithm [diakses pada 8 Desember 2023]

[5] https://hackernoon.com/revolutionizing-decision-making-with-python-and-slime-molds-journey-into-world-of-natural-computing [diakses pada 8 Desember 2023]

[6] https://www.sci.news/biology/slime-mold-problems-linear-time-06759.html [diakses pada 8 Desember 2023]

[7] https://www.hindawi.com/journals/tswj/2014/487069/ [diakses pada 8 Desember 2023]

[8] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9838547/ [diakses pada 8 Desember 2023]

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, 11 Desember 2023

Muhamad Rafli Rasyiidin (13522088)