MAKALAH PEMODELAN MATEMATIKA

OPTIMASI JUMLAH DAN LOKASI *WAREHOUSE* MENGGUNAKAN *GENETIC ALGORITHM*

Kelompok 7

Nadya Tjindra	2016710001
Leo Gunawan	2016710003
Geraldy Suryahartanto	2016710013
Yonathan Yeremy Budiman	2016710023
Al-Vinda Tania N.A.P	2016710025
Felix Tandiono	2016710028

4 November 2019

PROGRAM STUDI MATEMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2019

DAFTAR ISI

DAFTAR ISI iii

DAFTAR GAMBAR v

DAFTAR TABEL vii

1 PENDAHULUAN 1
- 1.1 Latar Belakang Masalah . . . 1
- 1.2 Rumusan Masalah . . . 1
- 1.3 Tujuan . . . 2
- 1.4 Manfaat . . . 2
- 1.5 Batasan masalah . . . 2
- 1.6 Metodologi . . . 3

2 DASAR TEORI 5
- 2.1 Genetic Algorithm . . . 5

3 HASIL DAN PEMBAHASAN 7
- 3.1 Bagian 1 . . . 7
- 3.2 Bagian 2 . . . 13
- 3.3 Bagian 3 . . . 16
- 3.4 Bagian 4 . . . 18

4 KESIMPULAN DAN SARAN 21
- 4.1 Kesimpulan . . . 21
- 4.2 Saran . . . 21

DAFTAR PUSTAKA 23

DAFTAR GAMBAR

2.1 Ilustrasi Alur Langkah *Genetic Algorithm* 6

3.1 Gambar Peta Pengiriman Diunduh Dari Situs UPS 7
3.2 Gambar Peta Pengiriman Hanya Dengan Daerah Berwarna Kuning 8
3.3 Daerah Pengiriman 1 Hari Dengan 15 *Warehouse* 9
3.4 Daerah Pengiriman 1 Hari Dengan 20 *Warehouse* 10
3.5 Peta Pengiriman 1 Hari Generasi Terbaik 13 Jumlah *Warehouse* 11
3.6 Peta Pengiriman Kode Gambar 0389 12
3.7 Visualisasi Pajak Pada Negara Bagian Amerika Serikat 14
3.8 Gambar Peta Pengiriman 1 Hari Peta Dengan Kode Gambar 0389 15

DAFTAR TABEL

3.1 Hasil *Genetic Algorithm* dengan 300 iterasi 9
3.2 Hasil *Genetic Algorithm* dengan 1000 Iterasi 10
3.3 Hasil Perhitungan Area *Overlap* Terhadap Generasi Terbaik Jumlah *Warehouse* 10 Sampai 15 dengan 1000 Iterasi 11
3.4 Kode Gambar dan *Zipcode* dari Solusi Penempatan *Warehouse* yang Dipilih 12
3.5 Pajak Penjualan Pada Setiap Negara Bagian Amerika Serikat 13
3.6 Pajak Pada Setiap *Warehouse* dari Solusi Bagian 1 14
3.7 Hasil Optimasi Lokasi *Warehouse* Solusi Bagian 1 Terhadap Pajak 15
3.8 Kode Gambar dan *Zipcode* dari Solusi Penempatan *Warehouse* yang Dipilih Dengan Mempertimbangkan Pajak 16
3.9 Pajak Penjualan Pakaian Pada Setiap Negara Bagian Amerika Serikat . . . 17
3.10 Pajak Pada Setiap *Warehouse* dari Solusi Bagian 2 18
3.11 Kode Gambar dan *Zipcode* dari Solusi Penempatan *Warehouse* yang Dipilih Dengan Mempertimbangkan Pajak Pakaian 18
3.12 Kode Gambar dan *Zipcode* dari Solusi Paling Optimal 19

BAB 1

PENDAHULUAN

1.1 Latar Belakang Masalah

Persaingan dalam dunia bisnis dapat dikatakan ketat pada era perdagangan modern, terlebih pada era globalisasi dimana persaingan dilakukan secara lokal dan global. Perusahaan harus memiliki sistem manajemen dan pemasaran yang baik agar bisnisnya berkembang. Pasar adalah kegiatan perekonomian yang memungkinkan pembeli dan penjual untuk saling bertukar jenis barang, jasa, ataupun informasi dengan menggunakan uang sebagai alat transaksinya. Pasar memungkinkan terjadinya perdagangan, distribusi, dan alokasi sumber daya dalam masyarakat. Salah satu wujudnya adalah toko dimana konsumen dapat menemukan barang ataupun jasa yang mereka butuhkan. Kegiatan menjual produk melalui toko ini dapat dilakukan salah satunya dengan dua cara, yaitu toko *online* atau toko konvensional (fisik). Perusahaan bisa saja membuka salah satu jenis toko atau membuka kedua jenis toko secara.

Pada kasus ini, suatu perusahaan memiliki toko konvensional yang berlokasi di New Hampshire, Amerika Serikat. Perusahaan tersebut ingin memperluas bisnisnya dengan membuka toko *online*. Toko *online* menawarkan kenyamanan berbelanja dari tempat apapun tanpa mendatangi tokonya secara fisik. Namun, toko *online* memiliki beberapa kendala yang harus diperhatikan, yaitu pengiriman dan perpajakan. Pengiriman yang cepat dan terpercaya merupakan suatu keharusan, sehingga perusahaan menginginkan pengiriman 1 hari terhadap seluruh daerah Amerika Serikat. Penempatan *warehouse* pada negara-negara bagian akan mempermudah dan mempercepat pengiriman di daerah tersebut. Namun, *warehouse* yang terlalu banyak akan merugikan perusahaan karena biaya dan pajak yang diberlakukan. Oleh karena itu, perusahaan harus mencari solusi optimal dari jumlah dan lokasi penempatan *warehouse* agar menutupi daerah Amerika Serikat sebanyak mungkin dengan pengiriman 1 hari dan dengan total pajak ditanggung yang rendah.

1.2 Rumusan Masalah

Rumusan masalah dari makalah ini adalah:

1. Bagaimana menentukan jumlah dan lokasi *warehouse* agar waktu pengiriman barang hanya 1 hari?

2. Bagaimana menentukan jumlah dan lokasi *warehouse* yang paling tepat agar biaya pajak dapat diminimalkan?

3. Bagaimana menganalisa dampak perubahan jumlah dan lokasi *warehouse* berdasarkan peningkatan penjualan pakaian?

1.3 Tujuan

Tujuan dari makalah ini adalah:

1. Menentukan jumlah dan lokasi *warehouse* agar waktu pengiriman barang hanya 1 hari untuk seluruh daerah di Amerika.
2. Menentukan jumlah dan lokasi *warehouse* yang paling tepat agar biaya pajak dapat diminimalkan.
3. Menentukan jumlah dan lokasi *warehouse* yang paling tepat agar biaya pajak pakaian dapat diminimalkan.
4. Menentukan jumlah dan lokasi *warehouse* yang paling tepat secara keseluruhan, baik diterapkan pajak maupun tidak.

1.4 Manfaat

Manfaat dari makalah ini adalah:

1. Mempelajari metode baru dalam memproses data menggunakan *Genetic Algorithm*.
2. Menerapkan teknologi dalam bidang matematika untuk menyelesaikan suatu permasalahan nyata.
3. Memodelkan sebuah masalah lalu mempresentasikan hasil yang didapatkan untuk mudah dipahami.

1.5 Batasan masalah

Berdasarkan data yang diperoleh, masalah dibatasi agar mempermudah proses pemodelan dan penarikan kesimpulan dari masalah tersebut. Batasan masalah yang dimaksud antara lain:

1. Jasa pengiriman yang digunakan adalah *United Parcel Service* (UPS) dan menggunakan jalir darat.
2. Seluruh peta pengiriman yang digunakan adalah peta yang diunduh dari situs resmi UPS pada September 2019 dan dianggap akurat.
3. Negara bagian Alaska, Puerto Rico dan Hawaii tidak tidak termasuk dalam pengiriman.
4. Setiap orang memiliki peluang beli yang sama, sehingga pembelian pada seluruh daerah Amerika dianggap rata dan sama.
5. Berat, luas, dan harga barang dianggap tidak berpengaruh.
6. Pajak ditanggung oleh perusahaan.
7. Pengiriman dianggap selalu lancar untuk sampai tujuan tanpa keterlambatan atau masalah.
8. Hasil dari *Genetic Algorithm* dianggap sebagai solusi yang paling mendekati.

1.6 Metodologi

Metode penelitian yang digunakan dalam makalah ini adalah *Genetic Algorithm*. Pertama, dilakukan pengunduhan semua gambar peta pengiriman yang terdapat pada situs UPS dan membuang peta pengiriman Alaska, Puerto Rico dan Hawaii. Kemudian, dilakukan perhitungan *fitness value* pada setiap gambar. Lalu, dengan menggunakan *Genetic Algorithm*, dapat dicari jumlah *warehouse* yang harus ditempatkan agar memaksimalkan pengiriman menjadi 1 hari di seluruh daerah Amerika Serikat. Proses pengunduhan, perhitungan piksel, pengolahan gambar, dan *Genetic Algorithm* dilakukan menggunakan bahasa pemrograman *Python3*.

BAB 2

DASAR TEORI

2.1 Genetic Algorithm

Genetic Algorithm [6] adalah algoritma yang mencerminkan proses seleksi alam di mana individu yang cocok dipilih untuk reproduksi agar menghasilkan keturunan dari generasi berikutnya. *Genetic Algorithm* merupakan metode penyelesaian yang bersifat heuristik, dimana metode ini merupakan teknik yang dirancang untuk memecahkan masalah yang mengabaikan apakah solusi yang didapat sudah optimal, namun biasanya teknik ini menghasilkan solusi yang baik dan dapat memecahkan masalah dengan lebih sederhana. Bentuk sederhana dari *Genetic Algorithm* melibatkan tiga jenis operator yaitu: seleksi, *Crossover* (titik tunggal), dan mutasi. Dalam seleksi operator ini memilih dua kromosom terbaik sebagai induk dalam populasi untuk reproduksi. Dalam *crossover*, operator memilih secara acak lokus (elemen) dan mengganti sub baris sebelumnya dalam kromosom dan kedua kromosom akan menghasilkan dua keturunan. Dalam mutasi, operator secara acak menukar beberapa kromosom dengan kromosom yang lebih baik yang telah dihasilkan.

Kromosom dalam populasi *Genetic Algorithm* biasanya memiliki bentuk barisan bit . Setiap lokus dalam kromosom memiliki dua kemungkinan alel: 0 dan 1. Setiap kromosom dapat dianggap sebagai titik pencarian solusi kandidat. *Genetic Algorithm* memproses populasi dari kromosom dengan menggantikan satu populasi dengan populasi lain secara berturut-turut. *Genetic Algorithm* paling sering membutuhkan fungsi *fitness* $f(x)$ yang menentukan nilai kecocokan atau *fitness value* bagi setiap kromosom di populasi. *Fitness value* dari sebuah kromosom bergantung pada seberapa baik kromosom mengatasi masalah.

Langkah kerja dari metode *Genetic Algorithm* yaitu:

1. *Genetic Algorithm* dimulai dengan populasi acak dari n 1−bit kromosom (kandidat solusi terhadap suatu masalah).

2. Menghitung *fitness value* $f(x)$ dari setiap kromosom x dalam populasi.

3. Mengulang langkah berikut sampai n keturunan telah terbentuk:

   (a) Memilih sepasang kromosom induk dari populasi. Kemungkinan dari seleksi akan meningkatkan fungsi dari *fitness value*. Pemilihan selesai dengan penggantian, yang berarti kromosom yang sama dapat dipilih lebih dari satu kali untuk menjadi induk.

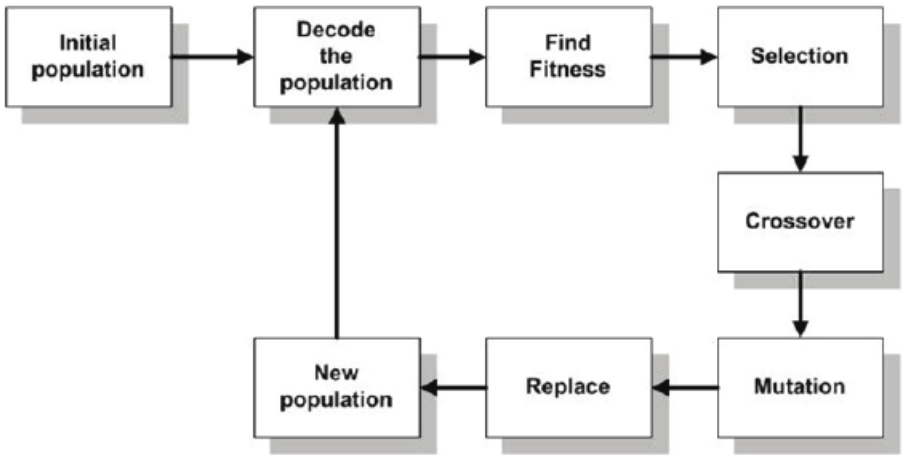
   (b) Dengan peluang p_c (peluang *crossover* atau *crossover* rate), pasangan induk disilangkan pada titik yang dipilih secara acak untuk membentuk dua keturunan. Jika tidak ada persilangan, dua keturunan yang sama persis dengan induk masing-masing akan terbentuk. Tingkat persilangan (*crossover rate*)

didefinisikan sebagai peluang kedua induk saling bertukar lokus pada satu titik. Terdapat versi persilangan banyak titik dari *Genetic Algorithm* yang mana tingkat persilangan untuk sepasang induk adalah jumlah titik dimana persilangan berlangsung).

(c) Melakukan mutasi terhadap dua keturunan pada setiap lokus dengan kemungkinan p_m (probabilitas mutasi atau tingkat mutasi). Kemudian, kromosom yang dihasilkan ditempatkan pada populasi baru. Jika n ganjil, satu anggota baru dapat dibuang secara acak.

4. Mengganti populasi saat ini dengan populasi baru.

5. Melakukan pengulangan langkah 2.



Gambar 2.1: Ilustrasi Alur Langkah *Genetic Algorithm*
Sumber: www.researchgate.net

Gambar 2.1 memberikan ilustrasi alur *Genetic Algorithm* yang sudah dibahas. *Genetic Algorithm* dimulai dengan populasi awal, lalu mencari *fitness value*, kemudian seleksi, *crossover*, mutasi, penggantian individu dalam populasi, mendapatkan populasi baru, kemudian kembali menghitung *fitness value* sampai jumlah generasi yang diinginkan didapat. Sifat *Genetic Algorithm* yang heuristik memungkinkan untuk mendapatkan generasi terbaik yang berbeda pada data dan alur yang sama pada pengulangan algoritma.

BAB 3

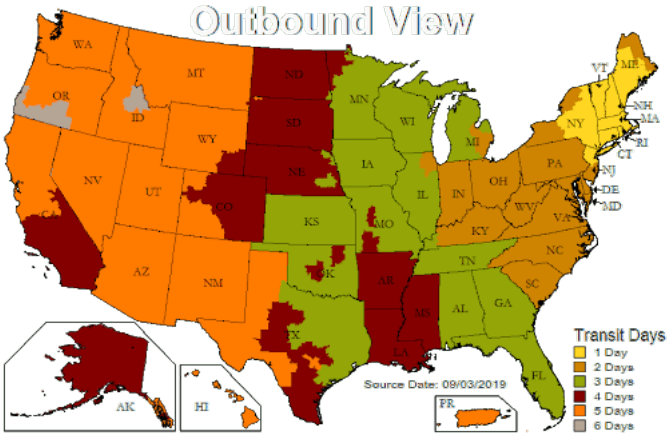
HASIL DAN PEMBAHASAN

Pada bab ini, Bagian 1 akan membahas penentuan jumlah dan lokasi *warehouse* agar waktu pengiriman barang hanya 1 hari untuk seluruh daerah di Amerika Serikat, Bagian 2 akan membahas penentuan jumlah dan lokasi *warehouse* yang paling tepat agar biaya pajak dapat diminimalkan, Bagian 3 akan membahas penentuan jumlah dan lokasi *warehouse* yang paling tepat agar biaya pajak pakaian dapat diminimalkan, dan Bagian 4 akan membahas penentuan jumlah dan lokasi *warehouse* yang paling tepat secara keseluruhan, baik diterapkan pajak maupun tidak.

3.1 Bagian 1

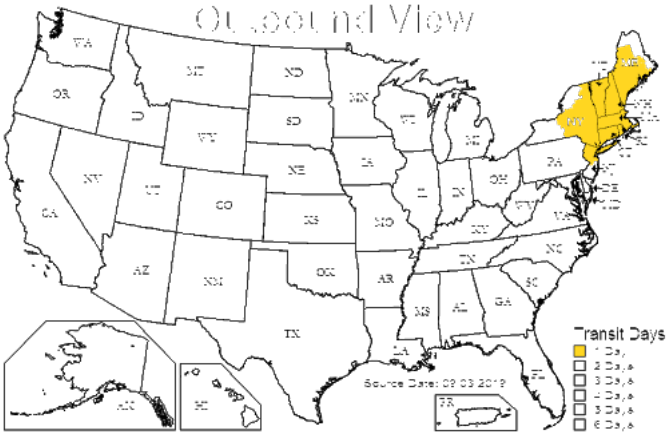
Data gambar yang digunakan dapat diambil dan diunduh dari situs resmi dari UPS yaitu www.ups.com/maps/. UPS adalah jasa pengiriman paket yang terpusat di Amerika Serikat dan digunakan sebagai jasa untuk pengiriman pada masalah ini. *Zipcode* atau kode pos yang kelompok kami dapatkan berjumlah 40.508. Sistem pada situs UPS mengelompokkan beberapa kode pos dalam satu daerah menjadi satu gambar, dikarenakan lokasinya yang saling mendekati sehingga pengiriman dari daerah tersebut akan sama satu sama lain, dan menghasilkan gambar yang sama. Hal ini mengakibatkan hanya 1.389 foto yang terunduh dari situs UPS, dengan penamaan maps_0001 sampai maps_1389. Karena Puerto Rico, Alaska, dan Hawaii tidak dianggap, maka data foto berkurang menjadi 1380 data.

Berikut adalah salah satu gambar yang terunduh dari situs UPS, dengan warna-warna pada daerah negara bagian menunjukkan durasi pengiriman paket dari sumber. Didapatkan luas daerah dari negara Amerika Serikat pada gambar adalah 301.664 piksel.



Gambar 3.1: Gambar Peta Pengiriman Diunduh Dari Situs UPS

Gambar diolah sehingga hanya perbatasan daerah yang berwarna hitam dan luas daerah berwarna kuning, yang menunjukkan daerah pengiriman satu hari, yang diambil dari gambar.



Gambar 3.2: Gambar Peta Pengiriman Hanya Dengan Daerah Berwarna Kuning

Oleh karena sulit untuk menentukkan sebenarnya darimana paket dikirim, kelompok kami memilih satu kode pos yang aproksimasi berada di tengah daerah berwarna kuning untuk merepresentasikan pusat pengiriman. Sehingga, satu gambar dianggap merepresentasikan satu *warehouse*.

Fitness value dari suatu himpunan merepresentasikan besar proporsi daerah negara Amerika Serikat yang tertutup oleh piksel kuning, yang berarti tertutup pengiriman satu hari, adalah:

$$f_1 = \frac{p_y}{p_t}$$

dengan p_y adalah jumlah piksel pada gambar yang berwarna kuning dan $p_t = 301664$ adalah jumlah piksel pada gambar yang merepresentasikan luas daerah negara Amerika Serikat. Langkah pengolahan data menggunakan *Genetic Algorithm* adalah sebagai berikut:

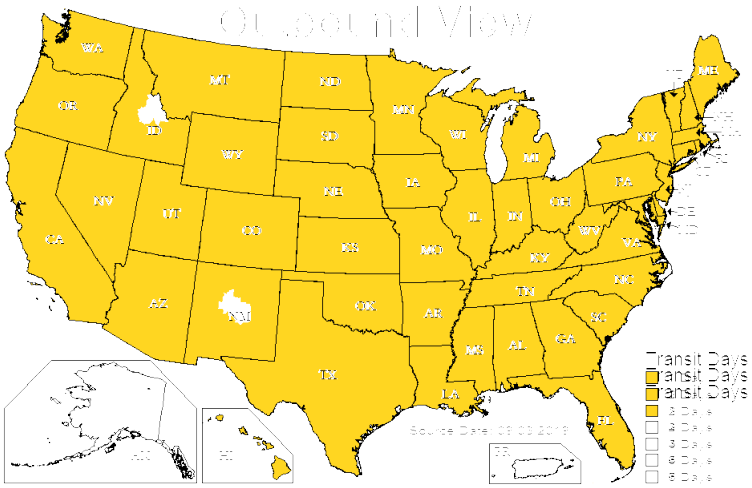
1. Mengambil 10 himpunan dengan masing-masing w jumlah kode gambar diambil dari populasi data.
2. Menghitung *fitness value* dari setiap himpunan.
3. Mengurutkan *fitness value* yang sudah dihitung dari yang terbesar sampai yang terkecil.
4. Dari 10 himpunan, 2 himpunan dengan *fitness value* paling tinggi disimpan dan 8 himpunan sisanya dibuang.
5. 2 himpunan terbaik yang disimpan akan saling bersilangan dan bermutasi dengan populasi pada data untuk menggantikan 8 himpunan yang sudah dibuang.
6. Mengulang iterasi sebanyak n kali untuk menghasilkan n generasi.

Pertama-tama dilakukan $n = 300$ iterasi untuk menentukkan selang jumlah *warehouse* yang optimal.

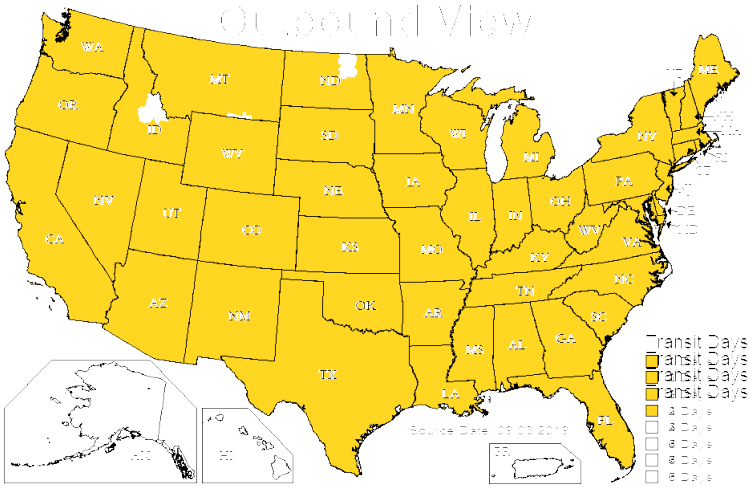
Jumlah *warehouse* (w)	Daerah Tertutup (%)
3	60,54517608995439
5	81,04778826774159
10	93,82624376790071
15	98,241752413281
20	98,39722340086984
25	98,59446271348254
30	98,87921661185956

Tabel 3.1: Hasil *Genetic Algorithm* dengan 300 iterasi

Tabel 3.1 menunjukkan bahwa semakin tinggi jumlah *warehouse*, semakin besar persentase daerah yang tertutup. Semakin tinggi persentase daerah tertutup, maka semakin luas daerah yang dapat dijangkau untuk 1 hari pengiriman. Hasil menunjukkan bahwa hasil daerah tertutup oleh 20 *warehouse*, 25 *warehouse*, dan 30 *warehouse* tidak mengalami peningkatan yang signifikan, maka dapat diasumsikan bahwa penutupan daerah tidak akan mencapai 100% dan hanya maksimal di sekitar 98,8%. Selisih ini mungkin dikarenakan ada ketidakakuratan pada piksel yang diolah di setiap gambar. Dengan asumsi kesalahan dan ketidakakuratan tersebut, dapat dikatakan daerah tertutup 98,8% sudah menutupi seluruh bagian gambar Amerika Serikat.



Gambar 3.3: Daerah Pengiriman 1 Hari Dengan 15 *Warehouse*



Gambar 3.4: Daerah Pengiriman 1 Hari Dengan 20 *Warehouse*

Tabel 3.1 memperlihatkan bahwa selisih daerah tertutup dengan 15 *warehouse* dan 20 *warehouse* sekitar 0.1%. Gambar 3.4 menunjukkan bahwa Gambar 3.3 memiliki daerah yang tidak tertutup kuning yang berbeda, namun persentase perbedaannya sangat kecil. Kelompok kami berpendapat bahwa kurang layak untuk menambah 5 *warehouse* hanya untuk menutup daerah sebesar 0.1%. Dapat disimpulkan bahwa 15 *warehouse* lebih optimal jika dibandingkan dengan 20 *warehouse*.

Berikutnya, akan dilakukan iterasi $n = 1000$ terhadap *warehouse* yang berjumlah 10 sampai 15 untuk mendapatkan solusi yang lebih optimal.

Jumlah *Warehouse* (w)	Daerah Tertutup (%)
10	97,97357324705632
11	97,97357324705632
12	97,69312877903893
13	98,73766839927867
14	98,73269597963297
15	98,42937838124536

Tabel 3.2: Hasil *Genetic Algorithm* dengan 1000 Iterasi

Tabel 3.2 memperlihatkan bahwa hasil daerah tertutup memiliki selisih antara satu sama lain yang sangat kecil. Oleh karena itu, sulit untuk menentukkan jumlah *warehouse* yang benar-benar optimal jika hanya melihat dari persentase daerah tertutup. Selanjutnya, akan dilakukan analisa lebih lanjut dengan menghitung area *overlap* atau daerah tumpang tindih.

Area *overlap* (daerah tumpang tindih) adalah daerah pada peta dimana piksel kuning saling menimpa satu sama lain, yang menunjukkan bahwa ada daerah pengiriman 1 hari yang berasal dari 2 atau lebih *warehouse* yang berbeda. Faktor ini menjadi indikasi bahwa hasil tidak efektif, dikarenakan penempatan *warehouse* yang tidak optimal mengakibatkan ada daerah yang ditutupi oleh 2 atau lebih *warehouse* berbeda secara bersamaan. area *overlap* memberikan kemungkinan bahwa mungkin saja ada solusi lain yang lebih optimal dengan jumlah *warehouse* yang lebih sedikit untuk menutupi seluruh daerah Amerika Serikat tanpa ada daerah yang tumpang tindih.

Area *overlap* dihitung untuk hasil generasi terbaik dari jumlah *warehouse* 10 sampai

15 yang sudah menjalani 1000 iterasi.

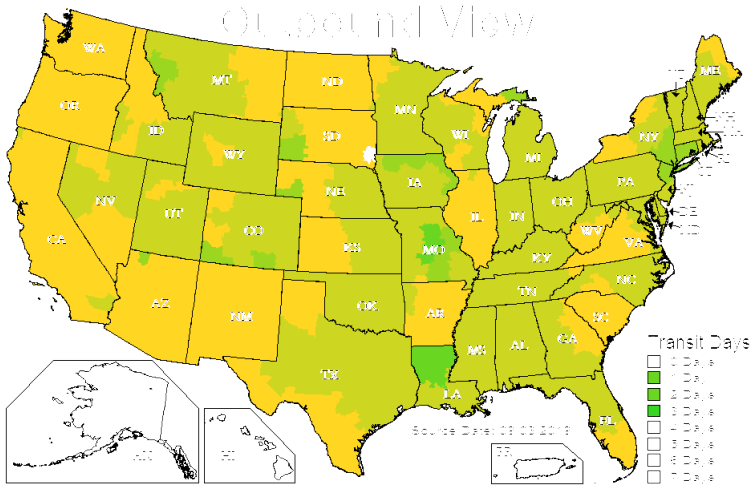
Jumlah *Warehouse* (w)	Overlap (%)
10	50,15547098758885
11	55,30557176196033
12	41,24124854142357
13	55,26877585658215
14	48,979659488702665
15	48,979659488702665

Tabel 3.3: Hasil Perhitungan Area *Overlap* Terhadap Generasi Terbaik Jumlah *Warehouse* 10 Sampai 15 dengan 1000 Iterasi

Tabel 3.3 menunjukkan bahwa area *overlap* untuk jumlah *warehouse* 10 sampai 15 bervariasi tanpa pola tertentu menunjukkan sifat *Genetic Algorithm* yang belum tentu memberikan hasil generasi terbaik bahkan setelah iterasi yang cukup banyak. Sifat ini dikarenakan pada setiap proses iterasi *Genetic Algorithm* dilakukan pengacakan data dan inisialisasi sehingga hasil tidak menjamin mendapatkan solusi optimal dari seluruh kombinasi gen. Bahkan, hasil *Genetic Algorithm* untuk populasi yang sama, gen yang sama, dan jumlah generasi yang sama dapat menghasilkan hasil yang berbeda.

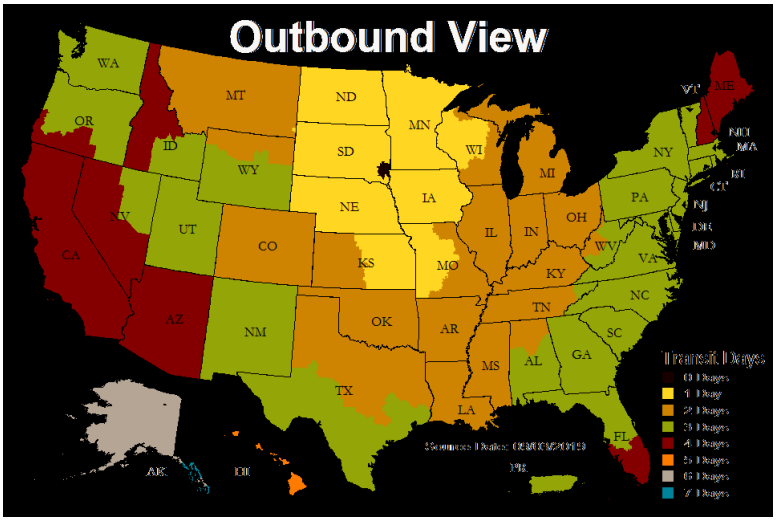
Hasil daerah tertutup dengan hasil area *overlap* tidak memberikan hasil yang jelas untuk jumlah *warehouse* yang paling optimal. Dengan demikian, kelompok kami memutuskan untuk membandingkan langsung hasil gambar 10 *warehouse* dengan 15 *warehouse*. Salah satu faktor yang paling menentukan adalah kepadatan penduduk di daerah yang tidak tertutup oleh piksel kuning. Meskipun daerah tertutup tidak bisa menutupi 100% dari daerah Amerika Serikat, kelompok kami masih memastikan bahwa daerah yang tidak tertutup tersebut merupakan daerah yang tidak padat penduduk, maka penduduk yang tidak dapat dilayani jumlahnya relatif sedikit.

Berdasarkan daerah tertutup, area *overlap*, dan kepadatan penduduk pada daerah yang tidak tertutup, kelompok kami memutuskan bahwa jumlah *warehouse* 13 adalah yang paling optimal. Berikut adalah gambar dari daerah tertutup *warehouse* 13



Gambar 3.5: Peta Pengiriman 1 Hari Generasi Terbaik 13 Jumlah *Warehouse*

Daerah pada gambar 3.5 yang tidak tertutup adalah pada negara bagian South Dakota.



Gambar 3.6: Peta Pengiriman Kode Gambar 0389

Gambar 3.6 menunjukkan bahwa daerah yang berwarna hitam adalah daerah pengiriman 0 hari. Daerah yang berwarna hitam tersebut menyerupai bagian daerah yang sama pada daerah putih gambar 3.5. Hal ini menunjukkan sesungguhnya peta 3.5 tertutup didaerah putih tersebut dikarenakan algoritma tidak menghitung daerah berwarna hitam yang merupakan pengiriman 0 hari.

Oleh karena itu, jika dilihat dari gambar asli dan gambar daerah tertutup, dapat disimpulkan bahwa generasi terbaik dari 13 *warehouse* hampir menutup seluruh daerah Amerika Serikat. Penempatan 13 *warehouse* tersebut adalah kode-kode gambar 'maps_0389.png', 'maps_0403.png', 'maps_1128.png', 'maps_1288.png', 'maps_0571.png', 'maps_1076.png', 'maps_0592.png', 'maps_0369.png', 'maps_0748.png', 'maps_1220.png', 'maps_0014.png', 'maps_0806.png', dan 'maps_0642.png'. Untuk memberikan penempatan *warehouse* yang lebih akurat, kelompok kami mengambil satu *zipcode* untuk setiap kode gambar, dengan aproksimasi titik tengah pada setiap peta pengiriman. *Zipcode* didapatkan dari situs worldpostalcode.com/united-states/.

Kode Gambar	Negara Bagian	*Zipcode*
maps_0389	South Dakota	57103
maps_0403	Montana	59462
maps_1128	West Virginia	24016
maps_1288	Utah	84111
maps_0571	Utah	84022
maps_1076	South Carolina	29209
maps_0592	New Mexico	88417
maps_0369	Maine	55008
maps_0748	Kansas	67035
maps_1220	New Mexico	87011
maps_0014	New Hampshire	3243
maps_0806	California	95815
maps_0642	Washington	99336

Tabel 3.4: Kode Gambar dan *Zipcode* dari Solusi Penempatan *Warehouse* yang Dipilih

3.2 Bagian 2

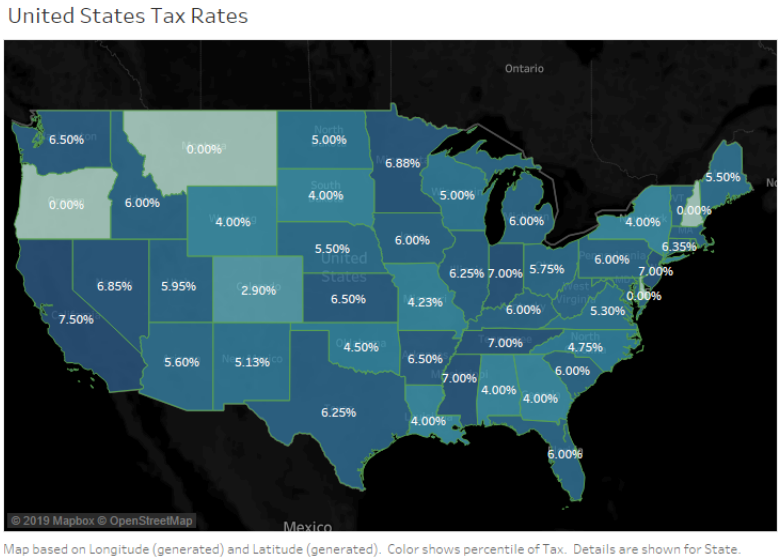
Selanjutnya akan di optimasi jumlah *warehouse* dengan mempertimbangkan pajak yang berlaku di setiap negara bagian. Pajak diberlakukan ketika pengiriman ke negara bagian dimana *warehouse* tersebut berada. Data pajak seluruh negara bagian di Amerika Serikat didapatkan dari situs http://www.tax-rates.org/taxtables/sales-tax-by-state. Besar pajak tersebut diurutkan dari yang terbesar hingga yang terkecil, sebagai berikut:

Nomor	Negara Bagian	Pajak (%)
1	California	7,50
2	Indiana	7,00
3	Mississippi	7,00
4	New Jersey	7,00
5	Rhode Island	7,00
6	Tennessee	7,00
7	Minnesota	6,88
8	Nevada	6,85
9	Arkansas	6,50
10	Kansas	6,50
11	Washington	6,50
12	Connecticut	6,35
13	Illinois	6,25
14	Massachusetts	6,25
15	Texas	6,25
16	Florida	6,00
17	Idaho	6,00
18	Iowa	6,00
19	Kentucky	6,00
20	Maryland	6,00
21	Michigan	6,00
22	Pennsylvania	6,00
23	South Carolina	6,00
24	Vermont	6,00
25	West Virginia	6,00

Nomor	Negara Bagian	Pajak
26	Utah	5,95
27	District of Columbia	5,75
28	Ohio	5,75
29	Arizona	5,60
30	Maine	5,50
31	Nebraska	5,50
32	Virginia	5,30
33	New Mexico	5,13
34	North Dakota	5,00
35	Wisconsin	5,00
36	North Carolina	4,75
37	Oklahoma	4,50
38	Missouri	4,23
39	Alabama	4,00
40	Georgia	4,00
41	Louisiana	4,00
42	New York	4,00
43	South Dakota	4,00
44	Wyoming	4,00
45	Colorado	2,90
46	Delaware	0,00
47	Montana	0,00
48	New Hampshire	0,00
49	Oregon	0,00

Tabel 3.5: Pajak Penjualan Pada Setiap Negara Bagian Amerika Serikat

dan dengan visualiasi sebagai berikut:



Gambar 3.7: Visualisasi Pajak Pada Negara Bagian Amerika Serikat

Negara bagian yang mengalami area *overlap* sesuai gambar 3.5, dimana daerahnya seluruhnya berwarna hijau muda, dianggap memiliki pajak 0%. Hal ini dikarenakan daerah yang *overlap* dapat dikirim dari 2 *warehouse* berbeda. Misalkan negara bagian Wyoming dan Idaho mengalami area *overlap* dan Wyoming harus mengirimkan barang untuk pelanggan di negara bagian tersebut, maka pengiriman akan terkena pajak. Untuk menghindari pajak, barang dapat dikirim dari Idaho. Negara bagian yang mengalami area *overlap* adalah IA (Iowa), OK (Oklahoma), LA (Los Angeles), MS (Minnesota), AL (Alabama), TN (Tennessee), KY (Kentucky), IN (Indiana), OH (Ohio), MI (Michigan), PA (Pennsylvania), NJ (New Jersey), DE (Delaware), MD (Maryland), CT (Connecticut), RI (Rhode Island), MA (Massachusetts), NH (New Hampshire), dan VT (Vermont). Maka, negara-negara bagian tersebut dianggap memiliki pajak 0%.

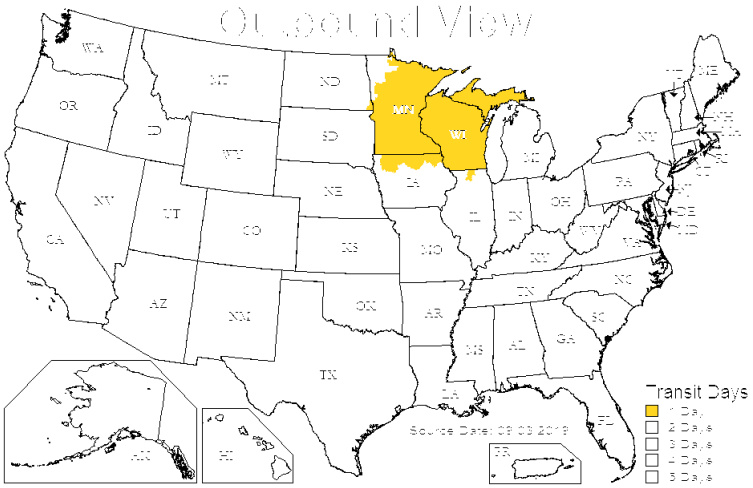
Pajak dari setiap *warehouse* pada solusi Bagian 1 sesuai 3.4 adalah:

Kode Gambar	Negara Bagian	Pajak I (%)
maps_0389	South Dakota	4
maps_0403	Montana	0
maps_1128	West Virginia	6
maps_1288	Utah	5,95
maps_0571	Utah	5,95
maps_1076	North Carolina	6
maps_0592	New Mexico	5,13
maps_0369	Maine	6,88
maps_0748	Kansas	6,5
maps_1220	New Mexico	5,13
maps_0014	New Hampshire	0
maps_0806	California	7,5
maps_0642	Washington	6,5

Tabel 3.6: Pajak Pada Setiap *Warehouse* dari Solusi Bagian 1

Tabel 3.6 memberikan setiap pajak yang dikenakan terhadap pengiriman setiap *warehouse* kepada negara bagiannya sendiri. Jika dijumlahkan, total dari pajak adalah 65,64%.

Untuk mengurangi efek dari pajak terhadap penempatan *warehouse*, kelompok kami melakukan optimasi dengan cara menggeser lokasi-lokasi *warehouse* ke negara bagian sebelahnya yang saling berdekatan. Dengan menggeser *warehouse* dekat ke perbatasan antara negara-negara bagian, pergeseran kecil tersebut tidak memberikan perubahan signifikan terhadap daerah pengiriman, tapi memberikan perubahan yang besar terhadap besar pajak ditanggung.



Gambar 3.8: Gambar Peta Pengiriman 1 Hari Peta Dengan Kode Gambar 0389

Sebagai contoh, pada gambar 3.8 yang merupakan peta pengiriman 1 hari untuk maps_0389, terlihat bahwa titik tengah daerah berwarna kuning adalah diantara negara bagian Maine dan Wisconsin. Maine memiliki pajak 6,88% sementara Wisconsin memiliki pajak 5%. Penempatan *warehouse* pada negara bagian Wisconsin yang dekat dengan perbatasan Maine akan memberikan daerah pengiriman 1 hari yang sama dan memberikan pajak yang lebih rendah. Hal yang sama diberlakukan terhadap setiap kode gambar pada solusi Bagian 1. Hasil dari optimasi lokasi *warehouse* solusi Bagian 1 terhadap pajak adalah:

Kode Gambar	Negara Bagian	Pajak II (%)
maps_0389	Iowa	0
maps_0403	Montana	0
maps_1128	Virginia	5,3
maps_1288	Wyoming	4
maps_0571	Utah	5,95
maps_1076	North Carolina	4,75
maps_0592	New Mexico	5,13
maps_0369	Wisconsin	5
maps_0748	Oklahoma	0
maps_1220	New Mexico	5,13
maps_0014	New Hampshire	0
maps_0806	Nevada	6,85
maps_0642	Oregon	0

Tabel 3.7: Hasil Optimasi Lokasi *Warehouse* Solusi Bagian 1 Terhadap Pajak

Tabel 3.7 memberikan total dari pajak sebesar 42,11%. Terlihat bahwa besar pajak menurun sebesar 23,53% dari 65,64%. Metode optimasi yang dilakukan terbukti menurunkan besar pajak secara signifikan agar tidak merugikan bisnis. Setelah perubahan penempatan lokasi *warehouse*, satu *zipcode* untuk setiap kode gambar dipilih ulang, dengan aproksimasi titik tengah pada setiap peta pengiriman. *Zipcode* didapatkan dari situs worldpostalcode.com/united-states/.

Kode Gambar	Negara Bagian	*Zipcode*
maps_0389	Iowa	51246
maps_0403	Montana	59462
maps_1128	Virginia	25911
maps_1288	Wyoming	82930
maps_0571	Utah	84022
maps_1076	North Carolina	28338
maps_0592	New Mexico	88417
maps_0369	Wisconsin	54013
maps_0748	Oklahoma	74653
maps_1220	New Mexico	87011
maps_0014	New Hampshire	3243
maps_0806	Nevada	89431
maps_0642	Oregon	97838

Tabel 3.8: Kode Gambar dan *Zipcode* dari Solusi Penempatan *Warehouse* yang Dipilih Dengan Mempertimbangkan Pajak

3.3 Bagian 3

Bagian ini akan membahas optimasi lokasi *warehouse* dengan mempertimbangkan pajak pakaian yang berlaku di setiap negara bagian. Sama seperti Bagian 2, pajak diberlakukan ketika pengiriman ke negara bagian dimana *warehouse* tersebut berada. Data pajak pakaian seluruh negara bagian di Amerika Serikat didapatkan dari data www.comap.com/highschool/contests/himcm/2016_Files/Sales_Tax_Rate.pdf. Data pajak tersebut diurutkan dari yang terbesar hingga yang terkecil, sehingga didapatkan hasil:

Nomor	Negara Bagian	Pajak (%)
1	California	7.50
2	Indiana	7.00
3	Mississippi	7.00
4	Tennessee	7.00
5	Nevada	6.85
6	Arkansas	6.50
7	Kansas	6.50
8	Washington	6.50
9	Connecticut	6.35
10	Illinois	6.25
11	Texas	6.25
12	Florida	6.00
13	Idaho	6.00
14	Iowa	6.00
15	Kentucky	6.00
16	Maryland	6.00
17	Michigan	6.00
18	South Carolina	6.00
19	West Virginia	6.00
20	Utah	5.95
21	District of Columbia	5.75
22	Ohio	5.75
23	Arizona	5.60
24	Maine	5.50
25	Nebraska	5.50

Nomor	Negara Bagian	Pajak
26	Virginia	5.30
27	New Mexico	5.13
28	North Dakota	5.00
29	Wisconsin	5.00
30	North Carolina	4.75
31	Oklahoma	4.50
32	Missouri	4.23
33	Alabama	4.00
34	Georgia	4.00
35	Louisiana	4.00
36	South Dakota	4.00
37	Wyoming	4.00
38	Colorado	2.90
39	New Jersey	0.00
40	Rhode Island	0.00
41	Minnesota	0.00
42	Massachusetts	0.00
43	Pennsylvania	0.00
44	Vermont	0.00
45	New York	0.00
46	Delaware	0.00
47	Montana	0.00
48	New Hampshire	0.00
49	Oregon	0.00

Tabel 3.9: Pajak Penjualan Pakaian Pada Setiap Negara Bagian Amerika Serikat

Perbedaan pajak penjualan biasa pada tabel 3.5 dan pajak penjualan pakaian pada 3.9 adalah bahwa pada beberapa negara bagaian, tidak diberlakukan pajak penjualan pakaian, contohnya negara bagian New Jersey, Rhode Island, Minnesota, dan sebagainya. Maka, untuk mendapatkan penerapan pajak pakaian terhadap model, digunakan optimasi terhadap solusi Bagian 2 berdasarkan 3.8.

Kode Gambar	Negara Bagian	Pajak III (%)
maps_0389	Iowa	0
maps_0403	Montana	0
maps_1128	Virginia	5,3
maps_1288	Wyoming	4
maps_0571	Utah	5,95
maps_1076	North Carolina	4,75
maps_0592	New Mexico	5,13
maps_0369	Wisconsin	0
maps_0748	Oklahoma	0
maps_1220	New Mexico	5,13
maps_0014	New Hampshire	0
maps_0806	Nevada	6,85
maps_0642	Oregon	0

Tabel 3.10: Pajak Pada Setiap *Warehouse* dari Solusi Bagian 2

Tabel 3.10 memberikan total pajak sebesar 37,11%. Penerapan pajak pakaian memberikan hasil yang lebih rendah dibandingkan penerapan pajak biasa, yaitu penurunan sebesar 5% dari 47,11%. Tidak terjadi pergeseran atau pemindahan lokasi *warehouse* dari solusi Bagian 2. Maka, solusi Bagian 3 adalah sama dengan solusi Bagian 2 sebagai berikut:

Kode Gambar	Negara Bagian	Zipcode
maps_0389	Iowa	51246
maps_0403	Montana	59462
maps_1128	Virginia	25911
maps_1288	Wyoming	82930
maps_0571	Utah	84022
maps_1076	North Carolina	28338
maps_0592	New Mexico	88417
maps_0369	Wisconsin	54013
maps_0748	Oklahoma	74653
maps_1220	New Mexico	87011
maps_0014	New Hampshire	3243
maps_0806	Nevada	89431
maps_0642	Oregon	97838

Tabel 3.11: Kode Gambar dan *Zipcode* dari Solusi Penempatan *Warehouse* yang Dipilih Dengan Mempertimbangkan Pajak Pakaian

3.4 Bagian 4

Berdasarkan hasil optimasi yang didapatkan dari Bagian 1 3.1, Bagian 2 3.2, dan Bagian 3 3.3, kelompok kami menentukan bahwa jumlah dan lokasi *zipcode* penempatan *warehouse* yang paling optimal adalah solusi Bagian 2, yang sama juga dengan solusi Bagian 3. Solusi yang paling optimal adalah sebagai berikut:

Kode Gambar	Negara Bagian	*Zipcode*
maps_0389	Iowa	51246
maps_0403	Montana	59462
maps_1128	Virginia	25911
maps_1288	Wyoming	82930
maps_0571	Utah	84022
maps_1076	North Carolina	28338
maps_0592	New Mexico	88417
maps_0369	Wisconsin	54013
maps_0748	Oklahoma	74653
maps_1220	New Mexico	87011
maps_0014	New Hampshire	3243
maps_0806	Nevada	89431
maps_0642	Oregon	97838

Tabel 3.12: Kode Gambar dan *Zipcode* dari Solusi Paling Optimal

Solusi yang paling optimal adalah menggunakan 13 jumlah *warehouse* dengan lokasi penempatan berdasarkan negara bagian dan *zipcode* 3.12. Jumlah *warehouse* yang adalah 13 memberikan daerah tertutup pengiriman 1 hari sebesar 98,73766839927867% yang dibahas pada 3.1 bahwa ia sebenarnya menutup hampir seluruh daerah. Besar area *overlap* 55,26877585658215% memberikan keuntungan bahwa daerah yang saling tumpang tindih memberikan kemudahan untuk menerapkan pajak 0% yang dibahas pada 3.2. Kemudian, berdasarkan pembahasan 3.2 dan 3.3, soluti optimal memberikan total pajak 42,11% pada penerapan pajak biasa dan 37,11% pada penerapan pajak pakaian.

BAB 4

KESIMPULAN DAN SARAN

4.1 Kesimpulan

Kesimpulan dari makalah ini adalah:

1. Jumlah dan lokasi *warehouse* yang paling optimal untuk pengiriman 1 hari terhadap seluruh daerah Amerika adalah 13 *warehouse* dengan *zipcode* 57103, 59462, 24016, 84111, 84022, 29209, 88417, 55008, 67035, 87011, 3243, 95815, dan 99336.

2. Jumlah dan lokasi *warehouse* yang paling optimal untuk pengiriman 1 hari dengan pajak penjualan minimal, yaitu sebesar 42,11%, terhadap seluruh daerah Amerika adalah 13 *warehouse* dengan *zipcode* 51246, 59462, 25911, 82930, 84022, 28338, 88417, 54013, 74653, 87011, 3243, 89431, dan 97838.

3. Jumlah dan lokasi *warehouse* yang paling optimal untuk pengiriman 1 hari dengan pajak penjualan pakaian minimal, yaitu sebesar 37,11%, terhadap seluruh daerah Amerika adalah 13 *warehouse* dengan *zipcode* 51246, 59462, 25911, 82930, 84022, 28338, 88417, 54013, 74653, 87011, 3243, 89431, dan 97838.

4. Jumlah dan lokasi *warehouse* yang paling optimal untuk pengiriman 1 hari secara keseluruhan terhadap seluruh daerah Amerika adalah 13 *warehouse* dengan *zipcode* 51246, 59462, 25911, 82930, 84022, 28338, 88417, 54013, 74653, 87011, 3243, 89431, dan 97838.

4.2 Saran

Saran dari makalah ini adalah:

1. Menggunakan data peta pengiriman yang terbaru dikarenakan situs UPS yang selalu memperbaharui data.

2. Menghitung daerah pengiriman yang berwarna hitam, yaitu daerah yang memiliki durasi pengiriman 0 hari.

3. Menggunakan *Genetic Algorithm* dalam menentukan jumlah dan lokasi *warehouse* yang tepat saat pajak dan pajak pakaian diberlakukan.

4. Menggunakan parameter-parameter tambahan seperti kepadatan penduduk, tingkat konsumsi, dan lain-lain dalam menyelesaikan permasalahan.

5. Menggunakan metode pemodelan lain yang tidak heuristik sehingga bisa didapatkan solusi optimal yang sesungguhnya.

DAFTAR PUSTAKA

[1] https://www.aggdata.com/free/united-states-zip-codes. *Complete List of United States Zip Codes*. AggData. Web. 25 Oct. 2019.

[2] https://www.comap.com/highschool/contests/himcm/2016_Files/Sales_Tax_Rate.pdf. *State Sales Tax Rates for 48 Continental US States*. Comap. Web. 25 Oct. 2019.

[3] http://www.whereig.com/usa/zipcodes. *U.S.A. Zip Codes by State*. WhereIG. Web. 25 Oct. 2019.

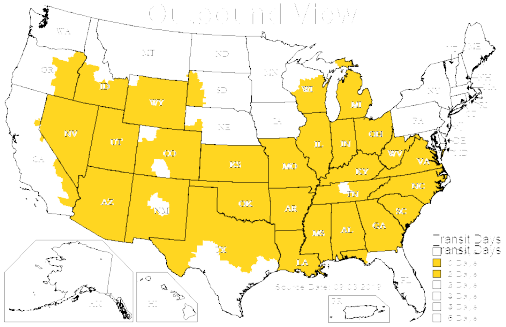
[4] http://www.tax-rates.org/taxtables/sales-tax-by-state. *Sales Tax Rates By State 2019*. Tax-rates.org. Web. 25 Oct. 2019.

[5] https://www.ups.com/maps. *United States Ground Time-in-Transit Maps*. The United Parcel Service. Web. 25 Sept. 2019.

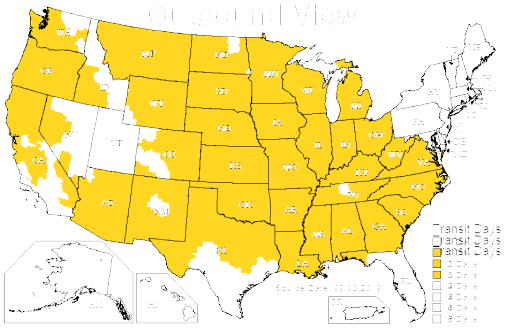
[6] M. Melanie. 1998. *An Introduction to Genetic Algorithms*. MIT Press. Cambridge.

LAMPIRAN A
HASIL *GENETIC ALGORITHM* 300 ITERASI

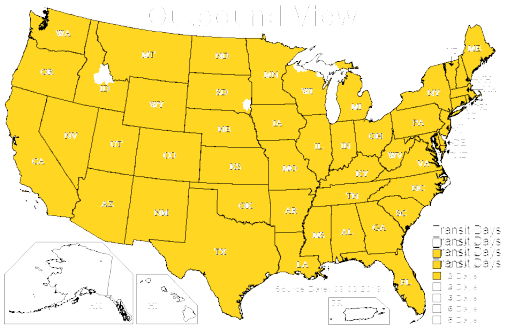
w =3, Area tertutup: 0.6054517608995439
'maps_0225.png', 'maps_1287.png', 'maps_0593.png'



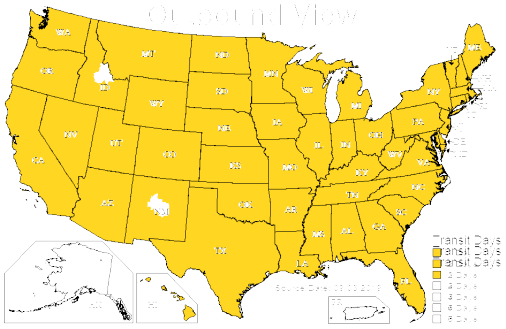
w =5 , Area tertutup: 0.8104778826774159
'maps_0593.png', 'maps_0673.png', 'maps_1128.png', 'maps_0353.png', 'maps_1287.png'



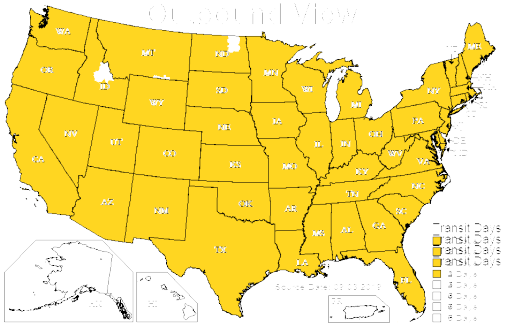
w =10, Area tertutup: 0.9382624376790071
'maps_0383.png', 'maps_1126.png', 'maps_1284.png', 'maps_1287.png', 'maps_1103.png', 'maps_0225.png', 'maps_0643.png', 'maps_1220.png', 'maps_0412.png', 'maps_0477.png'



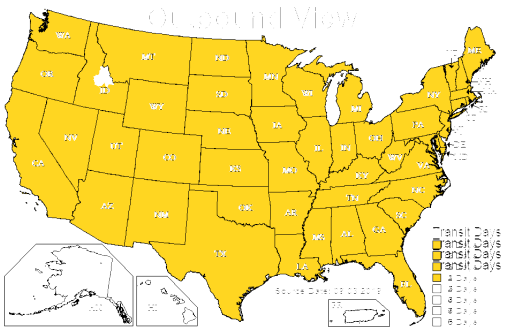
w =15, Area tertutup: 0.98241752413281
'maps_0214.png', 'maps_0365.png', 'maps_0731.png', 'maps_0642.png', 'maps_0353.png', 'maps_0874.png', 'maps_0794.png', 'maps_1305.png', 'maps_1288.png', 'maps_1280.png', 'maps_0082.png', 'maps_0383.png', 'maps_1008.png', 'maps_0408.png', 'maps_1219.png'



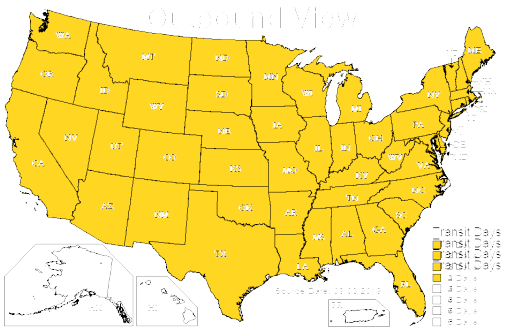
w =20, Area tertutup: 0.9839722340086984
'maps_0403.png', 'maps_0466.png', 'maps_1032.png', 'maps_0566.png', 'maps_1170.png', 'maps_0554.png', 'maps_0816.png', 'maps_0415.png', 'maps_0810.png', 'maps_0478.png', 'maps_0382.png', 'maps_0388.png', 'maps_0386.png', 'maps_0888.png', 'maps_1219.png', 'maps_1305.png', 'maps_1163.png', 'maps_0653.png', 'maps_0338.png', 'maps_1255.png'



w =25, Area tertutup: 0.9859446271348254
'maps_1336.png', 'maps_1097.png', 'maps_0030.png', 'maps_0369.png', 'maps_0403.png', 'maps_0391.png', 'maps_0675.png', 'maps_1239.png', 'maps_1120.png', 'maps_0385.png', 'maps_1284.png', 'maps_1032.png', 'maps_1288.png', 'maps_1218.png', 'maps_1163.png', 'maps_0404.png', 'maps_0561.png', 'maps_0592.png', 'maps_0471.png', 'maps_0554.png', 'maps_0231.png', 'maps_0935.png', 'maps_0407.png', 'maps_0383.png', 'maps_0214.png'

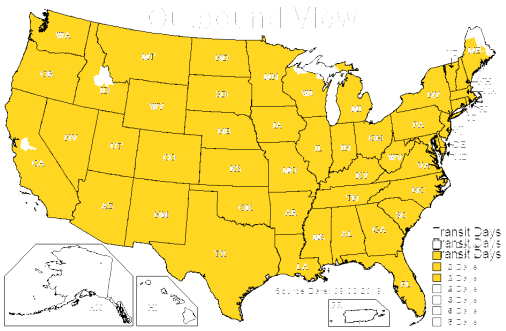


w =30, Area tertutup: 0.9887921661185956
'maps_0030.png', 'maps_1255.png', 'maps_0895.png', 'maps_1183.png', 'maps_0942.png', 'maps_0630.png', 'maps_1035.png', 'maps_0590.png', 'maps_0468.png', 'maps_0639.png', 'maps_1209.png', 'maps_1112.png', 'maps_0841.png', 'maps_1266.png', 'maps_0412.png', 'maps_0828.png', 'maps_0886.png', 'maps_0563.png', 'maps_0547.png', 'maps_0405.png', 'maps_0291.png', 'maps_0571.png', 'maps_0365.png', 'maps_0482.png', 'maps_1382.png', 'maps_0524.png', 'maps_0206.png', 'maps_0422.png', 'maps_1103.png', 'maps_0603.png'

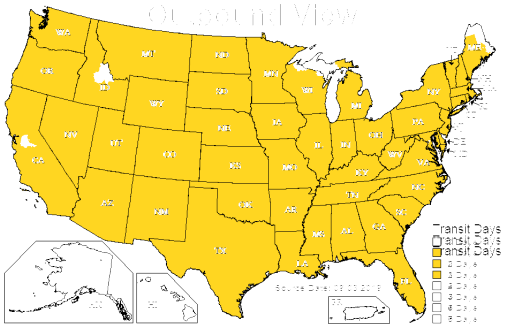


LAMPIRAN A
HASIL *GENETIC ALGORITHM* 1000 ITERASI

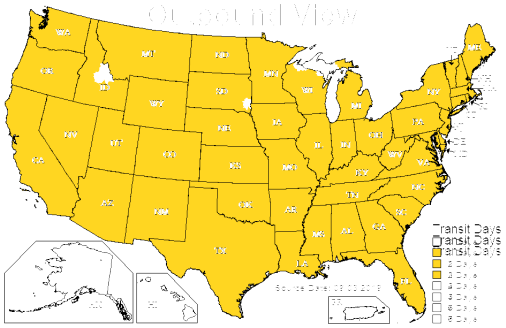
w =9, Area tertutup: 0.977249522647714
'maps_0620.png', 'maps_1130.png', 'maps_1220.png', 'maps_1103.png', 'maps_0353.png', 'maps_1288.png', 'maps_1241.png', 'maps_0405.png', 'maps_1032.png'



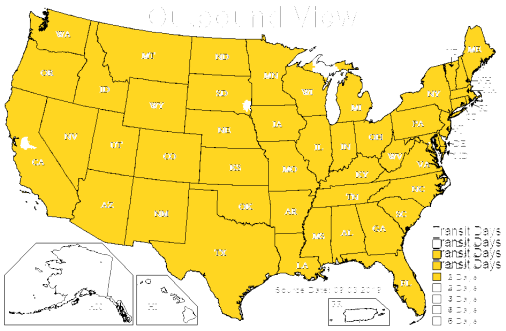
w =10, Area tertutup: 0.9797357324705632
'maps_0403.png', 'maps_1220.png', 'maps_0389.png', 'maps_1219.png', 'maps_0854.png', 'maps_1288.png', 'maps_1076.png', 'maps_0888.png', 'maps_0225.png', 'maps_1032.png'



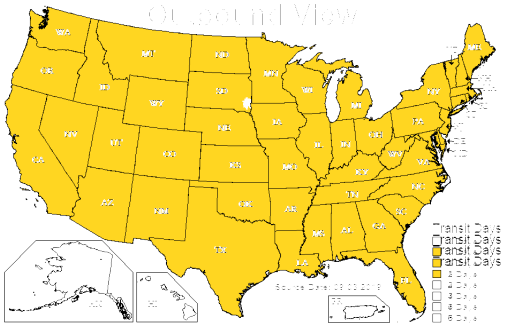
w =11 Area tertutup: 0.9797357324705632 'maps_1334.png', 'maps_0405.png', 'maps_1241.png', 'maps_1032.png', 'maps_1163.png', 'maps_1192.png', 'maps_0389.png', 'maps_0805.png', 'maps_1288.png', 'maps_1103.png', 'maps_0925.png'



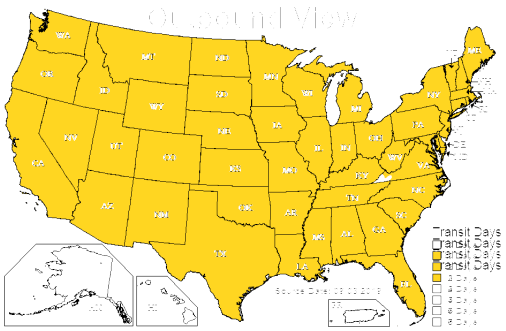
w =12 Area tertutup: 0.9769312877903893 'maps_1288.png', 'maps_0405.png', 'maps_0626.png', 'maps_0874.png', 'maps_0593.png', 'maps_1029.png', 'maps_1030.png', 'maps_1318.png', 'maps_0422.png', 'maps_1130.png', 'maps_0389.png', 'maps_0713.png'



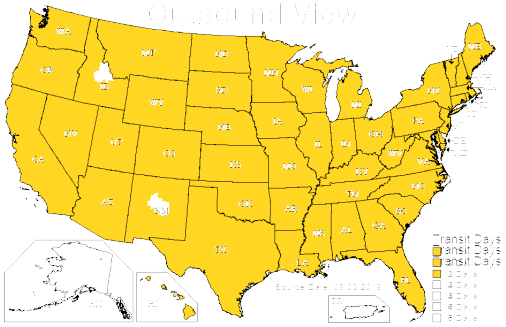
w =13 Area tertutup: 0.9873766839927867 'maps_0389.png', 'maps_0403.png', 'maps_1128.png', 'maps_1288.png', 'maps_0571.png', 'maps_1076.png', 'maps_0592.png', 'maps_0369.png', 'maps_0748.png', 'maps_1220.png', 'maps_0014.png', 'maps_0806.png', 'maps_0642.png'



$w = 14$ Area tertutup: 0.9873269597963297 'maps_1163.png', 'maps_0430.png', 'maps_0405.png', 'maps_0593.png', 'maps_1288.png', 'maps_0571.png', 'maps_0602.png', 'maps_1305.png', 'maps_0560.png', 'maps_1334.png', 'maps_0477.png', 'maps_0387.png', 'maps_1008.png', 'maps_0411.png'

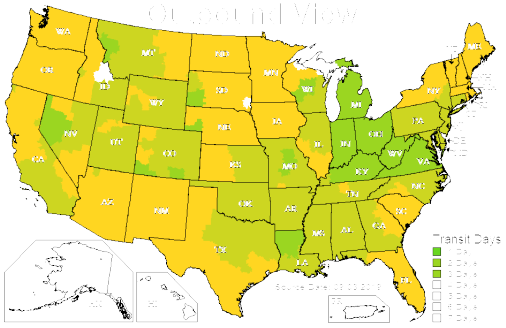


$w = 15$ Area tertutup: 0.9842937838124536 'maps_0382.png', 'maps_1287.png', 'maps_0593.png', 'maps_0353.png', 'maps_0214.png', 'maps_1163.png', 'maps_1305.png', 'maps_0365.png', 'maps_0673.png', 'maps_1076.png', 'maps_0632.png', 'maps_0403.png', 'maps_0912.png', 'maps_1283.png', 'maps_0649.png'

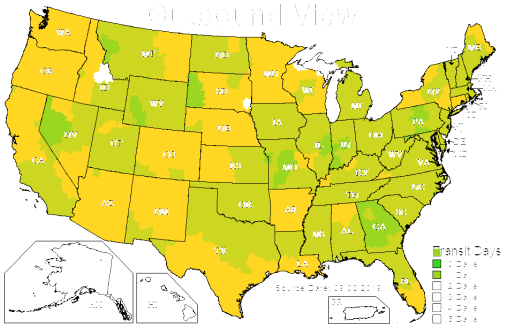


LAMPIRAN C
HASIL *OVERLAP*

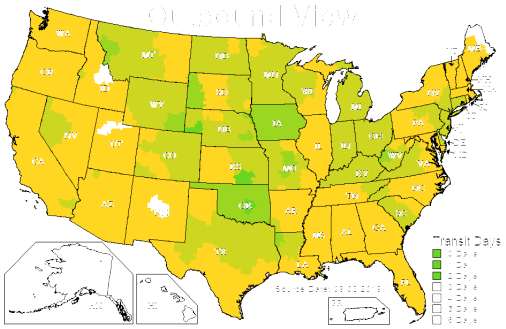
$w = 10$, Area *overlap*: 0.5015547098758885



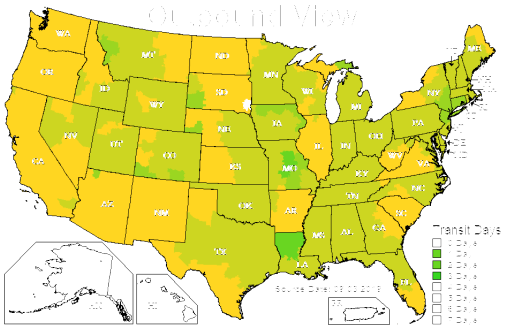
$w = 11$, Area *overlap*: 0.5530557176196033



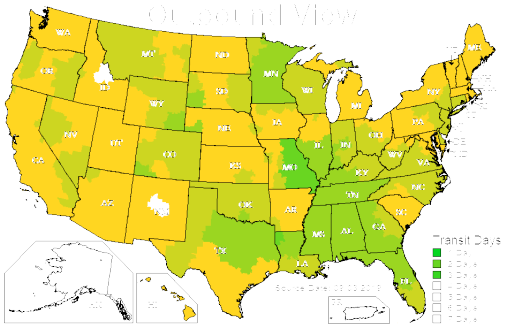
$w = 12$, Area *overlap*: 0.4124124854142357



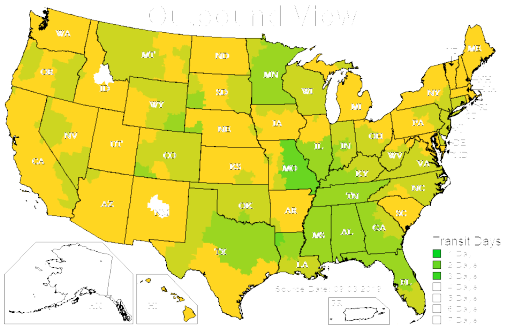
$w = 13$, Area *overlap*: 0.5526877585658215



$w = 14$, Area *overlap*: 0.48979659488702665



$w = 15$, Area *overlap*: 0.48979659488702665



LAMPIRAN D
CODING PYTHON

1. Mengunduh Gambar

```
import requests
import PIL
import PIL.Image
import urllib.request as urllib
import io
import numpy
import random
import csv

# Zip list
ZIP_LIST = []
with open('zipcodes.csv', 'r') as csvfile:
    linereader = csv.reader(csvfile)
    for row in linereader:
        zip = row[0]
        if len(zip) == 4:
            zip = "0" + zip
        if len(zip) == 3:
            zip = "00" + zip
        ZIP_LIST.append(zip)
print(len(ZIP_LIST))

# Fungsi untuk mengunduh gambar dari website UPS
def get_img(zip):
    url = requests.post("https://ups.com/using/services/servicemaps/maps25/
        "+zip+"_SEP19.gif", data={'zip': zip, 'stype': '0'})

    fd = urllib.urlopen(url)
    image_file = io.BytesIO(fd.read())
    img = PIL.Image.open(image_file)
    img = img.convert('RGBA')

    w, h = img.size
    pix = img.load()
    for x in range(w):
        for y in range(h):
            r, g, b, a = pix[x, y]
            if (r != 255 or g != 214 or b != 33) and (r != 0 or g != 0 or b
                != 0):
                pix[x, y] = (255, 255, 255, 0)
    img.save("images/" + zip + ".png")
    return img

i = 0
for zip in ZIP_LIST:
    try:
        get_img(zip)
    except:
        print("Image", i, "of", len(ZIP_LIST), "(Zip Code", zip + ") failed
            .")
    i += 1
    print("Downloaded image", i, "of", len(ZIP_LIST), "(ZipCode", zip + ")"
        )
```

2. Menghitung Piksel Pada Gambar

```
# Fungsi yang menghitung banyaknya piksel pada gambar, tanpa piksel
    transparan dan piksel hitam
count=0
img = PIL.Image.open('maps_0003.png').convert("RGBA")
w, h = img.size
pix = img.load()
for x in range(w):
    for y in range(h):
        r, g, b, a = pix[x, y]
        if (a!=0) and (r,g,b)!=(0,0,0) :
            count+=1

count
```

3. Melakukan Konversi Untuk Menyimpan Piksel Kuning dari Gambar

```
# Fungsi yang hanya mengambil piksel kuning dan hitam dari suatu gambar
def conv_img(zip):
    img = PIL.Image.open("images/" + zip).convert("RGBA")
    w, h = img.size
    pix = img.load()
    for x in range(w):
        for y in range(h):
            r, g, b, a = pix[x, y]
            if (r != 255 or g != 214 or b != 33) and (r != 0
               or g != 0 or b != 0):
                pix[x, y] = (255, 255, 255, 0)
    img.save("images_conv/" + zip)
    return img

for i in range(len(zips)):
    conv_img(zips[i])
```

4. Iterasi *Genetic Algorithm*

```
import PIL
import PIL.Image
import os
import pyperclip
import random
import pandas as pd
import numpy as np
import matplotlib

# Mengambil data-data gambar yang sudah ada
matplotlib.interactive(False)
zips = []
for file_name in os.listdir("images_conv/"):
    zips.append(file_name)

# Konstanta
QTY = []  # Jumlah warehouse
POP = 10  # Besar populasi
ITER = 1000  # Jumlah iterasi

TOTAL_PIXELS = 301664 # Total piksel luas daerah Amerika (tanpa border)
COLOR = (255, 214, 33,255)  # Warna kuning, menunjukkan pengiriman 1 hari
```

```
# Fungsi yang mengeluarkan data secara random
def gen_random_zips(q): #q= QTY jumlah warehouse
    random_zips = []
    for j in range(q):
        zip = top_zips[random.randint(0, len(top_zips) - 1)]
        i = np.array(zip)
        random_zips.append(zip)
    return random_zips

# Fungsi yang membaca gambar
def get_img(zip):
    img = PIL.Image.open("images_conv/" + zip)
    return img


# Fungsi yang menghitung berapa jumlah piksel dari suatu warna yang ada
    pada gambar
def count_color(img, color):
    colors = img.getcolors()
    pixels = None
    for tup in colors:
        if tup[1] == color:
            return tup[0]

for i in range(len(zips)):
    img = get_img(zips[i])
    if type(img)==None:
        print(zips[i])


# Fungsi yang menentukan fitness value dari suatu himpunan warehouse
def fitness(zips, show=False):
    comb_img = get_img(zips[0]).copy()
    #comb_img = pyperclip.copy(get_img(zips[0]))
    for i in range(len(zips)):
        img = get_img(zips[i]).convert("RGBA")
        [x,y]=img.size
        comb_img.paste(img, (0,0), img)
    if show == True:
        comb_img.show()
        comb_img.save('warehouse_'+STR+'.png')
    filled_pixels = count_color(comb_img, COLOR)
    #filled_pixels -= count_color(comb_img.crop((466, 213, 545, 352)),
        COLOR)
    # print("Filled pixels:", filled_pixels)
    del comb_img
    return filled_pixels / TOTAL_PIXELS

# Fungsi yang melakukan clone terhadap gen
def clone(array):
    new_array = []
    for elem in array:
        new_array.append(elem)
    return new_array

# Fungsi yang secara mengganti suatu kode gambar dengan kode gambar lain
    secara acak
def mutate(zips):
    new_zips = clone(zips)
    ind = random.randint(0, len(new_zips) - 1)
    new_zips[ind] = gen_random_zips(1)[0]
```

```
    return new_zips
gen = []
for i in range(POP):
    gen.append(gen_random_zips(QTY))

for i in range(ITER):
    gen = sorted(gen, key=fitness, reverse=True)
    print("Generation:", i, "Best:", fitness(gen[0]))
    for j in range(2, POP, 1):
        gen[j] = mutate(gen[j%2])

print("FINAL BEST:", fitness(gen[0], show=True))
print(gen[0])
```

5. Mengeluarkan Gambar Hasil Generasi Terbaik

```
import PIL
import PIL.Image
import numpy as np
import pandas as pd

# Himpunan kode gambar terbaik
zips=['HASIL GENERASI TERBAIK']
top_zips = []

# Mengambil gambar
def get_img(i):
  try:
    img = PIL.Image.open("images_conv/" + zips[i]).convert('RGBA')
    return img
  except:
    return False

base_img=None
colorup=np.array([255,235,124])
colordown=np.array([211,193,0])

# Fungsi untuk menumpuk gambar di atas satu sama lain, lalu mengubah area
    overlap menjadi warna hijau
def add_img(zip, img):
  width, height = img.size
  pix = img.load()
  count = 0
  overlap_count = 0
  b_pix = base_img.load()
  for x in range(width):
    for y in range(height):
      r,g,b,a= pix[x,y]
      if (r != 255 or g != 214 or b != 33) and (r != 0 or g != 0 or b != 0)
          :
        pix[x, y] = (255, 255, 255, 0)
      if (r == 255 and g == 214 and b == 33):
        count += 1
        b_r, b_g, b_b, b_a = b_pix[x,y]
        if (b_g == g and b_b == b):
          b_pix[x, y] = (b_r - 50, g, b)
          overlap_count += 1
        else:
          b_pix[x,y] = (r, g, b)
  print(zip, "Count:", count, "Overlap:", overlap_count)
  if count >= 6000:
```

```
    top_zips.append((zip, count))

# Fungsi yang memanggil gambar
for i in range(0,1389):
  img = get_img(i)
  if img != False:
    width, height = img.size
    pix = img.load()
    count = 0
    if base_img == None:
      base_img = img
      for x in range(width):
        for y in range(height):
          r,g,b,a= pix[x,y]
          if (r != 255 or g != 214 or b != 33) and (r != 0 or g != 0 or b !
              = 0):
            pix[x, y] = (255, 255, 255, 0)
          if (r == 255 and g == 214 and b == 33):
              count += 1
      print(i, ":", count-56)
    else:
      add_img(i, img)
      if count >= 6000:
        top_zips.append((i, count))
  else:
   print("Failed to open image of", i)

base_img.show()
print(top_zips)

while True:
  inp = input("add zip code or type q to quit: ")
  if inp == "q":
    break
  elif int(inp) != None:
    add_img(get_img(inp))
    base_img.show()
```

6. Menghitung Area *Overlap* Pada Gambar

```
import PIL
import PIL.Image
import numpy as np
import pandas as pd

# Data yang ingin di cek overlap
zips=['overlap_10.png','overlap_11.png', 'overlap_12.png', 'overlap_13.png'
    ,'overlap_14.png','overlap_15.png']

# Fungsi menghitung piksel yang tidak kuning dan tidak hitam, artinya
    piksel yang memiliki warna sekitar hijau
def count_overlap(zip):
    count=0
    img = PIL.Image.open(zip).convert("RGBA")
    w, h = img.size
    pix = img.load()
    for x in range(w):
        for y in range(h):
            r, g, b, a = pix[x, y]
            if (r,g,b)!=(255,214,33) and (r,g,b)!=(0,0,0) and a!=0:
                count += 1
```

```
    print(zip, "overlap", count/301664)

for zip in zips:
    count_overlap(zip)
```