Alternatif Penentuan Daerah yang Terkena Dampak Kabut Asap Menggunakan *Quickhull*

Studi Kasus pada Pulau Sumatra dan Kalimantan

Shevalda Gracielira – 13516134
Program Studi Sarjana Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516134@std.stei.itb.ac.id

***Abstrak*—Makalah ini memberikan alternatif dalam penentuan daerah yang terkena dampak kabut asap dengan program yang menggunakan algoritma *quickhull*. Fokus utama dari makalah ini adalah pulau Sumatra dan Kalimantan. Alternatif dalam penentuan ini berguna untuk membantu penanganan terhadap korban kabut asap yang harus ditindaklanjuti dengan cepat.**

Kata-kata kunci—*quickhull*, kabut asap, Sumatra, Kalimantan

I. Pendahuluan

Bencana kabut asap di Indonesia sudah tidak asing lagi. Setiap tahunnya pulau Sumatra dan Kalimantan selalu mempunyai berita yang meliput bagaimana tahun ini ada kabut asap akibat pembakaran lahan. Namun, sepertinya setiap kali ada berita mengenai kabut asap di media, dampak kabut asap di lapangan sudah jauh lebih mengkhawatirkan. Kabut asap nyatanya sudah menyebar ke berbagai daerah karena berbagai faktor seperti munculnya titik api baru atau asap yang dibawa oleh angin.

Sejauh ini, satelit pencitraan digunakan untuk mengetahui jangkauan kabut asap. Sayangnya, tidak semua lembaga mempunyai teknologi tersebut sehingga penanganan dari pemerintah dapat diperlambat dan kurang tepat sasaran karena kurangnya informasi tentang fakta di lapangan. Untuk mengetahui situasi lapangan, pemerintah sering kali harus melihat langsung ke daerah yang sudah terkena dampak kabut asap. Tentu saja, mengirimkan orang-orang untuk sekadar memeriksa situasi cukup berbahaya dengan pertimbangan dampak menghirup asap ketika turun ke lapangan.

Makalah ini mengeskplorasi suatu alternatif untuk mengetahui jangkauan kabut asap ketika sudah diketahui beberapa daerah yang sudah dipastikan terkena bencana kabut asap. Alternatif tersebut memanfaatkan algoritma *quickhull* untuk melihat daerah mana saja yang ikut terkena dampak kabut asap.

II. Landasan Teori

A. Algoritma *Divide-and-Conquer*

Algoritma *divide-and-conquer* adalah algoritma yang menggunakan prinsip rekursi *multi-branched*. Algoritma ini menguntungkan ketika dibutuhkan sebuah algoritma yang dapat memecahkan permasalahan kompleks, memberikan perkiraan yang cukup akurat, memiliki tingkat efisiensi yang tinggi dalam waktu dan memori, dan memungkinkan untuk dilakukan secara paralel.

Garis besar algoritma *divide-and-conquer* dapat dipecah menjadi tiga bagian:

1. *Divide*: suatu permasalahan dibagi menjadi beberapa permasalahan yang lebih kecil. Biasanya permasalahan ini dibagi dua secara rekursif.
2. *Conquer*: bagian-bagian permasalahan yang sudah tidak dapat dibagi diselesaikan per bagian sehingga permasalahan yang dipecahkan tidak sebanyak atau serumit permasalahan awal.
3. *Combine*: solusi yang telah ditemukan untuk setiap bagian-bagian kecil dikembalikan dan mungkin digabung sedemikian rupa untuk menemukan solusi global.

Kompleksitas algoritma tersebut dapat dinyatakan sebagai berikut:

$$O(\mathrm{n}) = \begin{cases} T(1), & \mathrm{n} = 1 \\ \mathrm{a}T\left(\frac{\mathrm{n}}{\mathrm{b}}\right) + f(\mathrm{n}), & \mathrm{n} > 1 \end{cases}$$

di mana *a* dan *b* dan konstanta yang bergantung pembagian permasalahan menjadi bagian yang lebih kecil. *f(n)* adalah kompleksitas penggabungan solusi permasalahan dari bagian-bagian permasalahan yang telah dipecah tadi.

Algoritma *divide-and-conquer* diimplementasikan dalam algoritma *mergesort*, *quicksort*, *tranversal* dalam pohon biner, perkalian bilangan besar, perkalian *Strassen's Matrix*, permasalahan pasangan terdekat, dan *convex-hull*.

B. *Convex-Hull* dan Algoritma *Quickhull*

C*onvex-hull* adalah himpunan *convex* paling sedikit yang mencakup suatu himpunan titik dalam suatu parameter. *Convex* sendiri adalah himpunan titik yang jika suatu pasang titik, misalkan *p* dan *q*, membentuk garis, tidak ada garis dari pasangan titik lain dalam himpunan yang mencakup titik pasangan *p* dan *q* yang dapat melewati garis yang dibentuk pasangan *p* dan *q*.

Algoritma untuk mencari *convex-hull* dapat dilakukan dengan pendekatan algoritma *brute force* dan algoritma *divide-and-conquer*. Pencarian *convex-hull* menggunakan pendekatan *brute force* mencari semua kemungkinan himpunan yang memenuhi syarat sebagai *convex-hull*. Biasanya pendekatan ini merupakan permasalahan kombitorial.

Pencarian *convex-hull* menggunakan pendekatan *divide-and-conquer*, sering kali disebut sebagai *quickull* karena mirip dengan *quick merge. Quickhull* membagi permasalahan menjadi dua bagian, bagian kiri dan bagian kanan. Kemudian setelah pembagian pertama, pemecahan permasalahan dilakukan secara rekursif.

Algoritma *quickhull* dapat digambarkan melalui *pseudocode* berikut:

```
points ← larik dari sekumpulan titik-titik

Mencari titik paling kiri dan titik paling kanan
dari points

left_point ← titik paling kiri

right_point ← titik paling kanan

Bagi titik-titik yang bukan left_point dan
right_point menjadi titik-titik di bagian kiri
dan kanan berdasarkan garis yang dibentuk
left_point dan  right_point

set_of_left_points ← titik-titik di bagian kiri
garis

set_of_right_points ← titik-titik di bagian
kanan garis

result_left ← findhull(set_of_left_points,
left_point, right_point)

result_right ← findhull(set_of_right_points,
right_point, left_point)

Result ← penggabungan result_left dan
result_right
```

Pseudocode dari fungsi rekursif findhull adalah sebagai berikut:

```
function findhull(result: array of points, left:
point, right: point) → array of points

    if result adalah larik kosong then

        → result

    else

        Cari titik yang letaknya paling jauh
        dari left dan right

        point_max ← titik yang letaknya paling
        jauh

        Cari titik-titik yang ada di bagian kiri
        dari garis left dan point_max

        Cari titik-titik yang ada di bagian kiri
        dari garis point_max dan right

        left_result ← larik dari titik-titik
        yang ada di bagian kiri garis left dan
        point_max

        right_result ← larik dari titik-titik
        yang ada di bagian kiri garis point_max
        dan right

        → gabungan dari hasil
        findhull(left_result, left, point_max)
        dan findhull(right_result, point_max,
        right)
```

C. Kabut Asap

Kabut asap, atau sering disebut *smog* (*smoke and fog*) dalam Bahasa Inggris, adalah sejenis polusi udara yang mempunyai ciri-ciri udara yang tampak berkabut dan udara yang berbau. Kabut asap dapat disebabkan oleh reaksi dari nitrogen oksida dan hidrokarbon yang terkena paparan cahaya matahari, pembakaran batu bara, emisi dari transportasi, hasil letusan gunung berapi, dan kebakaran hutan.

Dalam jangka pendek, kabut asap dapat menyebabkan pengurangan jarak pandang yang dapat membahayakan pengemudi darat maupun udara, iritasi pada mata dan tengorokan, dan memperparah gejala asma. Dalam jangka panjang, kabut asap dapat menyebabkan berkurangnya harapan hidup, menambah resiko Alzheimer, menyebabkan catat pada kandungan dan gangguan kesehatan yang berhubungan dengan kabut asap seperti kanker paru-paru, dan menyebabkan kematian prematur.

D. Bencana Kabut Asap di Indonesia

Kabut asap di Indonesia yang parah dihasilkan dari pembakaran hutan yang akan dijadikan lahan yang dipakai untuk produksi *pulp*, kertas, dan minyak. Pembakaran lahan ini kemudian dilakukan setiap tahun untuk panen sawit tahunan. Selain itu, musim kemarau juga memperparah polusi asap tersebut.

Di Indonesia, pengukuran kualitas polutan udara diatur oleh Keputusan Kepala Badan Pengendalian Dampak Lingkungan nomor KEP-107/KABAPEDAL/11/1997. Pada keputusan tersebut, dinyatakan untuk mengunakan Indeks Standar Pencemaran Udara (ISPU) untuk mengukur kualitas polusi udara. Parameter ISPU ditentukan oleh lima substansi, yaitu Partikulat (PM10), sulfur dioksida (SO_2), karbon monoksida (CO), ozon (O_3), dan nitrogen dioksida (NO_2). Masing-masing

substansi diukur setiap 24 jam, 8 jam, atau 1 jam. Kemudian, hasil rata-rata dari pengukuran tersebut diumumkan setiap 24 jam.

Berikut rentang indeks, kategori, dan kemungkinan dampak jika ISPU ada di rentang tersebut.

PENGARUH INDEKS STANDAR PENCEMAR UDARA UNTUK SETIAP PARAMETER PENCEMAR

Kategori	Rentang	Carbon Monoksida (CO)	Nitrogen (NO2)	Ozon O3	Sulfur Dioksida (SO2)	Partikulat
Baik	0-50	Tidak ada efek	Sedikit berbau	Luka pada Beberapa spesies tumbuhan akibat Kombinasi dengan SO2 (Selama 4 Jam)	Luka pada Beberapa spesies tumbuhan akibat kombinasi dengan O3 (Selama 4 Jam)	Tidak ada efek
Sedang	51 - 100	Perubahan kimia darah tapi tidak terdeteksi	Berbau	Luka pada Babarapa spesies tumbuhan	Luka pada Beberapa spesies lumbuhan	Terjadi penurunan pada jarak pandang
Tidak Sehat	101 - 199	Peningkatan pada kardiovaskularpada perokok yang sakit jantung	Bau dan kehilangan warna. Peningkatan reaktivitas pembuluh tenggorokan pada penderita asma	Penurunan kemampuan pada atlit yang berlatih keras	Bau, Meningkatnya kerusakan tanaman	Jarak pandang turun dan terjadi pengotoran debu di mana-mana
Sangat Tidak Sehat	200-299	Maningkatnya kardiovaskular pada orang bukan perokok yang berpanyakit Jantung, dan akan tampak beberapa kalemahan yang terlihat secara nyata	Meningkatnya sensitivitas pasien yang berpenyaklt asma dan bronhitis	Olah raga ringan mangakibatkan pengaruh parnafasan pada pasien yang berpenyaklt paru-paru kronis	Meningkatnya sensitivitas pada pasien berpenyakit asthma dan bronhitis	Meningkatnya sensitivitas pada pasien berpenyakit asthma dan bronhitis
Berbahaya	300 - lebih	Tingkat yang berbahaya bagi semua populasi yang terpapar				

Tabel 1 Pengaruh Indeks Standar Pencemar Udara untuk Setiap Parameter Pencemar
Sumber: Lampiran III Keputusan Kepala Badan Pengendalian Dampak Lingkungan nomor KEP-107/KABAPEDAL/11/1997

Kabut asap terparah di Indonesia terjadi pada tahun 1997, 2013, dan 2015. Kabut asap yang terjadi pada 2015 tercatat mencapai 983 dalam indeks kualitas udara di kota Pekanbaru.

III. Pemecahan Masalah

Dalam untuk mengetahui daerah-daerah yang kemungkinan besar terkena bencana kabut asap tanpa menggunakan citra dari satelit, penulis menggunakan sebuah program yang menggunakan *quickhull*.

Asumsi penulis adalah ketika diketahui beberapa daerah yang sudah terkena dampak kabut asap, ada kemungkinan besar daerah-daerah yang dikelilingi daerah tersebut juga terkena dampak tanpa harus dikonfirmasi dari penduduk daerah tersebut. Pada pencitraan satelit, tidak selalu diketahui lokasi-lokasi yang mungkin kabut asapnya tidak terlalu tebal namun sudah mulai memberi dampak pada penduduk daerah tersebut.

A. Cara Kerja Program

Program yang buat oleh penulis berdasarkan algoritma *quickhull*. Program yang dibuat oleh penulis menggunakan bahasa Python 3.

Pertama-pertama, pengguna akan diminta untuk memilih pulau apa yang akan dianalisis. Program sejauh ini baru dapat menganalisis untuk pulau Sumatra dan Kalimantan. Analisis sengaja dibuat terpisah untuk pulau Sumatra dan Kalimantan supaya studi kasus lebih detil.

Setelah pengguna menentukan pulau mana yang akan dianalisis, pengguna diminta untuk memasukkan kota-kota yang diketahui sudah terkena dampak kabut asap. Program sejauh ini baru dapat menerima ibukota-ibukota provinsi pada tiap pulau untuk merepresentasikan provinsinya. Untuk menentukan kota mana yang dipilih, pengguna diminta untuk memasukkan kode kota sebagai masukkan.

Untuk pulau Sumatra, berikut kode, kota, dan provinsi yang direpresentasikan kota tersebut:

Kode	Kota	Provinsi
0	Banda Aceh	Aceh
1	Medan	Sumatra Utara
2	Padang	Sumatra Barat
3	Pekanbaru	Riau
4	Jambi	Jambi
5	Palembang	Sumatra Selatan
6	Bengkulu	Bengkulu
7	Bandar Lampung	Lampung
8	Pangkal Pinang	Kepulauan Bangka Belitung
9	Tanjung Pinang	Kepulauan Riau

Tabel 2 Kode untuk kota-kota di pulau Sumatra yang digunakan oleh program
Sumber: penulis

Untuk pulau Kalimantan, berikut kode, kota, dan provinsi yang direpresentasikan kota tersebut:

Kode	Kota	Provinsi
0	Pontianak	Kalimantan Barat
1	Palangka Raya	Kalimantan Tengah
2	Banjarmasin	Kalimantan Selatan
3	Samarinda	Kalimantan Timur

Tabel 3 Kode untuk kota-kota di pulau Kalimantan yang digunakan oleh program
Sumber: penulis

Program sejauh ini mempunyai minimum tiga kota yang harus diinput dan batasan maksimalnya ada sebanyak jumlah kota untuk setiap pulau. Pembacaan akan diakhiri ketika penguna memasukkan kode -1.

Setelah pengguna sudah memasukkan nama-nama kota, program akan mengolah nama-nama tersebut menjadi koordinat dua dimensi yang sudah diatur oleh penulis. Koordinat untuk setiap kota diusahakan untuk semirip mungkin dengan letak di peta. Untuk detail koordinat yang diberikan untuk setiap kota pada pulau Sumatra dan Kalimantan dapat dilihat di Lampiran 1 dan Lampiran 2.

Dari hasil pengolahan koordinat di tahap sebelumnya, program akan mulai menerapkan algoritma *quickhull* terhadap koordinat-koordinat tersebut. Kode dari fungsi yang menggunakan algoritma *quickhull* dan fungsi findConvexHull untuk membantu algoritma *quickhull* tersebut dapat dilihat pada Lampiran 3 dan Lampiran 4.

Solusi dari algoritma *quickhull* tersebut akan ditampilkan dengan penggambaran *plot* yang disediakan oleh *library* matplotlib. Pada hasil penggambaran tersebut, akan terlihat daerah-daerah mana saja yang kemungkinan terkena bencana kabut asap. Kota-kota yang diinput dari pengguna akan ditandai dengan warna merah sedangkan kota-kota lain yang bukan termasuk masukan akan ditandai dengan warna biru.

B. Studi Kasus

Studi Kasus dibagi menjadi dua yaitu untuk pulau Sumatra dan pulau Kalimantan. Untuk setiap pulau terdapat dua kasus berbeda sehingga ada total empat kasus yang akan dianalisis.

Berikut daftar kota yang akan diuji untuk setiap studi kasus dan masukkan ke program:

1) Studi kasus 1 pulau Sumatra
   Medan, Pekanbaru, Tanjung Pinang, Padang, Bengkulu, dan Pangkal Pinang.

```
Pulau yang ingin dianalisa (sumatra/kalimantan):
sumatra
Masukkan kode daerah yang terkena kabut asap:
1
3
9
2
6
8
-1
```

2) Studi kasus 2 pulau Sumatra
   Banda Aceh, Bandar Lampung, Pangkal Pinang, Palembang.

```
Pulau yang ingin dianalisa (sumatra/kalimantan):
sumatra
Masukkan kode daerah yang terkena kabut asap:
1
7
8
5
-1
```

3) Studi kasus 1 pulau Kalimantan
   Pontianak, Palangka Raya, Banjarmasin

```
Pulau yang ingin dianalisa (sumatra/kalimantan):
kalimantan
Masukkan kode daerah yang terkena kabut asap:
0
1
2
-1
```

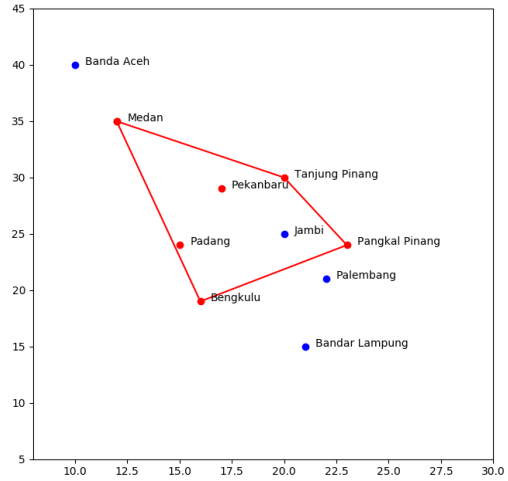
4) Studi kasus 2 pulau Kalimantan
   Palangka Raya, Banjarmasin, Pontianak, Samarinda

```
Pulau yang ingin dianalisa (sumatra/kalimantan):
kalimantan
Masukkan kode daerah yang terkena kabut asap:
1
2
0
3
-1
```

C. Hasil Program dan Analisis

Berikut hasil dari program yang telah dijalankan. Analisis yang dijelaskan adalah hasil analisis dari keluaran program. Variabel lain seperti jumlah titik api pada suatu kota diabaikan untuk analisis ini karena belum diperhitungkan dalam program.

1) Studi kasus 1 pulau Sumatra



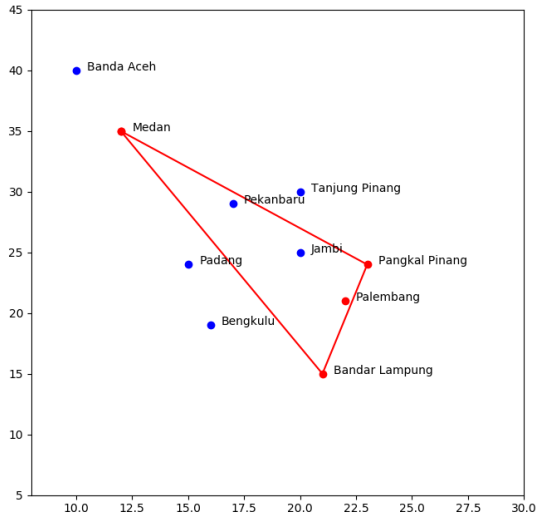
Gambar 1 Hasil dari studi kasus 1 untuk pulau Sumatra
Sumber: penulis

Pada hasil studi kasus tersebut, didapatkan bahwa kota-kota yang merupakan *convex-hull* adalah kota Medan, kota Tanjung Pinang, kota Pangkal Pinang, dan kota Bengkulu.

Kota Pekanbaru dan kota Padang tampak berada di dalam daerah *convex-hull*. Pengamatan ini secara tidak langsung memberikan gambaran bahwa kabut asap di kota Pekanbaru dan kota Padang seharusnya lebih parah dibanding kota-kota hasil *convex-hull*.

Satu hal menarik lain yang dapat diamati adalah bahwa kota Jambi, yang bukan merupakan salah satu kota yang diketahui terkena kabut asap, ternyata berada di dalam area *convex-hull*. Oleh karena itu, kota Jambi dapat disimpulkan juga mengalami bencana kabut asap juga.

2) Studi kasus 2 pulau Sumatra

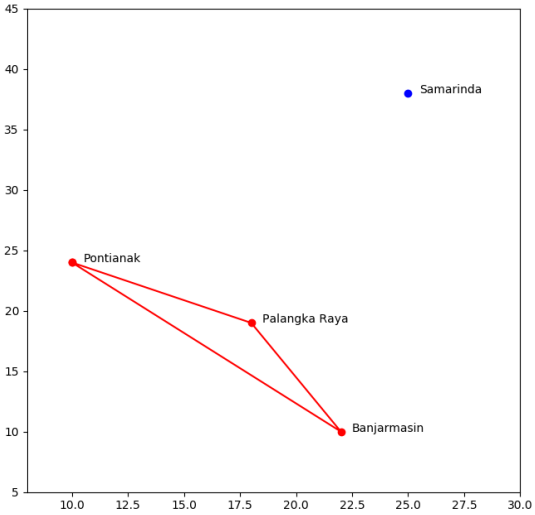


Gambar 2 Hasil dari studi kasus 2 untuk pulau Sumatra
Sumber: penulis

Hasil dari *convex-hull* adalah kota Medan, Pangkal Pinang, dan Bandar Lampung. Kota Jambi dan Pekanbaru terlihat berada di dalam area *convex-hull* tersebut. Namun, perbedaannya adalah kita dapat menyimpulkan bahwa kabut asap di kota Jambi

tertentunya lebih parah dari pada kota Pekanbaru. Hal ini tergambar dengan dua kota yang menjadi *convex-hull* dan satu kota yang tidak berada lebih dekat dengan kota Jambi disbanding Pekanbaru yang lebih dekat dengan hanya satu kota hasil *convex-hull*.

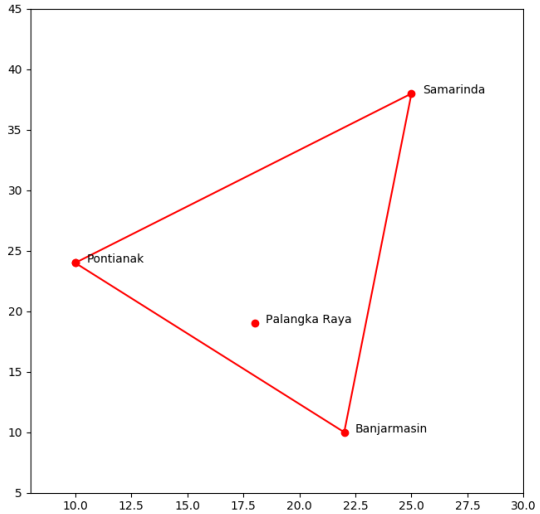
3) Studi kasus 1 pulau Kalimantan



Gambar 3 Hasil dari studi kasus 1 untuk pulau Sumatra
Sumber: penulis

Pada studi kasus ini, semua kota yang diketahui terkena dampak kabut asap menjadi hasil *convex-hull*. Namun, dari hasil *convex-hull* tersebut dapat diketahui bahwa kabut asap kemungkinan besar belum sampai ke kota Samarinda karena daerah *convex-hull* cukup jauh dengan lokas kota Samarinda.

4) Studi kasus 2 pulau Kalimantan



Gambar 4 Hasil dari studi kasus 1 untuk pulau Sumatra
Sumber: penulis

Pada studi kasus ini, di mana semua kota di Kalimantan diketahui telah mengalami bencana kabut asap, memang tidak mengejutkan bahwa hasil *convex-hull* adalah tiga kota terluar dari pulau Kalimantan. Namun, yang dapat diperhatikan dari hasil tersebut adalah bagaimana kota Palangka Raya, sebagai kota yang dikelilingi oleh tiga kota lainnya, menjadi kota yang terkena dampak kabut asap paling parah.

D. Analisis terhadap Hasil Keseluruhan

Program yang telah dibuat dapat menentukan kisaran daerah minimum yang merasakan dampak kabut asap. Dari hasil program, pengguna dapat mengamati kota-kota mana saja yang mungkin sudah terkena dampak kabut asap namun belum ada pemberitahuan bahwa daerah tersebut sudah terkena kabut asap. Selain itu, dari daerah hasil *convex-hull*, pengguna dapat mengetahui kota-kota mana saja yang merasakan dampak kabut asap lebih parah dibanding kota-kota lainnya.

Tentu saja, program dapat dikembangkan lebih jauh. Pada program yang dibuat, untuk setiap pulau, pengguna hanya dapat memilih ibukota setiap provinsi pada pulau tersebut. Hal ini tentu memiliki kelemahannya tersendiri seperti pada studi kasus pulau Kalimantan karena kota yang dapat diuji hanya ada lima. Permasalahan ini dapat diatasi dengan menambah titik daerah untuk setiap pulau. Lebih baik lagi jika titik-titik yang muncul pada peta tersebut adalah daerah-daerah yang memang terbukti sering terkena bencana kabut asap.

Algoritma yang digunakan pun belum dapat memperhitungkan faktor-faktor lain yang dapat mempengaruhi kabut asap seperti seberapa jumlah titik api pada suatu daerah dan cuaca iklim, seperti angin, pada saat analisis dilakukan. Hal ini dapat diatasi dengan menambah beberapa variabel yang dapat diperhitungkan juga oleh program sehingga hasil lebih merepresentasikan situasi sebenarnya.

IV. KESIMPULAN

Penanganan kabut asap sering kali terhambat karena kurangnya informasi di lapangan sehingga tindak lanjut dating terlambat. Keterlambatan informasi tersebut dapat disebabkan oleh teknologi yang tidak miliki semua pemerintah. Dengan algoritma *quickhull* dan informasi tentang beberapa daerah yang diketahui sudah terkena bencana kabut asap, maka mengetahui daerah-daerah lain yang mengalami kabut asap dan daerah yang merasakan dampak kabut asap paling parah dapat diatasi.

Algoritma *quickhull* sendiri belum cukup untuk memperkirakan situasi di lapangan secara akurat. Di sisi lain, algoritma tersebut cukup untuk memberikan gambaran umum untuk mengambil keputusan yang mendesak seperti dalam menindaklanjuti korban kabut asap.

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena tanpa rahmat-Nya, penulis tidak mungkin menyelesaikan makalah ini dengan baik. Penulis juga ingin berterima kasih kepada Bu Nur Ulfa Maulidevi, Pak Rinaldi Munir, dan Bu Masayu Leylia Khodra selaku dosen mata kuliah Strategi Algoritma. Penulis juga berterima kasih kepada teman-teman seperjuangan yang terus memberi semangat secara langsung maupun tidak langsung. Terakhir, penulis berterima kasih dan bersyukur kepada kedua orang tua penulis yang tidak henti-hentinya mendukung setiap harinya.

REFERENSI

[1] Levitin, A. (2012). *Introduction to the Design & Analysis of Algorithms*. Boston: Pearson Education.

[2] Munir, R. (2018). *Diktat Kuliah IF2211: Strategi Algoritma*. Bandung.

[3] Horowitz, E., Sahni, S., & Rajasekaran, S. (1998). *Computer Algorithms*. W. H. Freeman and Company.

[4] M. Frigo; C. E. Leiserson; H. Prokop (1999). "Cache-oblivious algorithms". *Proc. 40th Symp. on the Foundations of Computer Science*.

[5] Barber, C. B., Dobkin, D. P., & Huhdanpaa, H. (1996). The Quickhull Algorithm for Convex Hulls. *ACM Transactions on Mathematical Software*.

[6] Piazzesi, Gaia (2006). *The Catalytic Hydrolysis of Isocyanic Acid (HNCO) in the Urea-SCR Process*. ETH Zurich.

[7] Chris (2007). *Environmentalism in 1306*. Environmental Graffiti.

[8] *EPA Tools Available as Summer Smog Season Starts* (Press release). Boston, Massachusetts: United States Environmental Protection Agency. 30 April 2008.

[9] Causes and Effects of Smog. (2016, December 25). Retrieved from https://www.conserve-energy-future.com/smogpollution.php

[10] McCafferty, G., & Sater, T. (2015, September 18). Indonesian haze: Why it's everyone's problem. *CNN*. Retrieved from https://edition.cnn.com/2015/09/17/asia/indonesian-haze-southeast-asia-pollution/index.html

[11] Toxic air pollution nanoparticles discovered in the human brain. (2016, September 5). *Lancaster University*. Retrieved from http://www.lancaster.ac.uk/news/articles/2016/toxic-air-pollution-nanoparticles-discovered-in-the-human-brain/

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, 13 Mei 2018

Shevalda Gracielira 13516134

Lampiran 1: Implementasi algoritma *quickhull* dalam Python 3

```
def quickhull(point list):
    """
    :param point_list: titik-titik dari kota yang diketahui terkena dampak kabut asap
    :return: titik-titik kota yang menjadi convex-hull area kabut asap
    """
    point_list = sorted(point_list, key=itemgetter(0, 1))
    left_end = point_list[0]
    point list.remove(left end)
    right_end = point_list[len(point_list) - 1]
    point_list.remove(right_end)

    upper_points = []  # titik pada upper bound
    lower points = []  # titik pada lower bound
    for point in point list:
        side length = findSideAndDistance(point, left end, right end)
        if (side_length > 0):  # di bagian atas garis pembatas
            upper_points.append(point)
        elif (side_length < 0):  # di bagian bawah garis pembatas
            lower_points.append(point)

    upper result = findConvexHullPairs([], left end, right end, upper points)
    lower_result = findConvexHullPairs([], right_end, left_end, lower_points)

    result = deepcopy(upper_result)
    result.extend(lower_result)

    return result
```

Lampiran 2: Implementasi fungsi findConvexHullPairs dalam Python 3

```
def findConvexHullPairs(container, left_point, right_point, candidate_list):
    """
    :param container: hasil yang telah didapat dari hasil rekursif sebelumnya
    :param left_point: titik yang menjadi bagian kiri garis pembatas
    :param right_point: titik yang menjadi bagian kanan garis pembatas
    :param candidate list: titik-titik yang mungkin menjadi convex
    :return: solusi dari bagian yang telah dibagi sebelumnya
    """

    new_container = deepcopy(container)
    if len(candidate_list) == 0:  # jika sudah tidak ada lagi titik di sebelah kirinya
        new container.append([left point, right point])
    else:
        # Mencari titik paling jauh dari garis yang dibentuk left_point dan right_point
        max_candidate = []
        max_length = 0
        for candidate in candidate_list:
            distance = findSideAndDistance(candidate, left_point, right_point)
            if distance > max length:
                max_candidate = []
                max_candidate.append(candidate)
                max_length = distance
            elif distance == max_length:
                max candidate.append(candidate)
        if len(max candidate) == 1:
            pmax = max candidate[0]
        else:
            max_degree = 0
            for candidate in max_candidate:
                degree = findDegree(candidate, left_point, right_point)
                if degree > max_degree:
                    pmax = candidate
                    max degree = degree
        candidate_list.remove(pmax)

        # Mencari membuat candidate_list untuk sebelah kiri dan kanan
        left_candidate = []
        right candidate = []
        for candidate in candidate list:
            if findSideAndDistance(candidate, left_point, pmax) > 0:  # titik untuk sebelah kiri
                left_candidate.append(candidate)
            elif findSideAndDistance(candidate, pmax, right_point) > 0:  # titik untuk sebelah kanan
                right_candidate.append(candidate)

        left result = findConvexHullPairs(new container, left point, pmax, left candidate)
        right_result = findConvexHullPairs(new_container, pmax, right_point, right_candidate)

        # Mengabungkan hasil dari bagian kiri dan kanan
        new_container.extend(left_result)
        new_container.extend(right_result)
    return new_container
```

Lampiran 3: *Dictionary* dari kota-kota di pulau Sumatra dan Kalimantan, berserta koordinatnya

```
sumatra = [
    {'name': 'Banda Aceh', 'coordinate': [10, 40]},
    {'name': 'Medan', 'coordinate': [12, 35]},
    {'name': 'Padang', 'coordinate': [15, 24]},
    {'name': 'Pekanbaru', 'coordinate': [17, 29]},
    {'name': 'Jambi', 'coordinate': [20, 25]},
    {'name': 'Palembang', 'coordinate': [22, 21]},
    {'name': 'Bengkulu', 'coordinate': [16, 19]},
    {'name': 'Bandar Lampung', 'coordinate': [21, 15]},
    {'name': 'Pangkal Pinang', 'coordinate': [23, 24]},
    {'name': 'Tanjung Pinang', 'coordinate': [20, 30]},
]

kalimantan = [
    {'name': 'Pontianak', 'coordinate': [10, 24]},
    {'name': 'Palangka Raya', 'coordinate': [18, 19]},
    {'name': 'Banjarmasin', 'coordinate': [22, 10]},
    {'name': 'Samarinda', 'coordinate': [25, 38]},
]
```