Penerapan Algoritma Greedy dalam Membantu Penjadwalan Alat Pemberi Isyarat Lalu Lintas

Yakobus Iryanto Prasethio - 13520104
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: yakobusiryantoprasethio@gmail.com

***Abstract*— Jumlah kendaraan di seluruh dunia semakin hari semakin bertambah. Untuk memastikan keamanan dan kenyamanan dalam berkendara di jalan raya, maka dibuatlah sebuah alat yang dapat mengatur lalu lintas tanpa membutuhkan campur tangan manusia. Alat Pemberi Isyarat Lalu Lintas atau yang biasa disebut dengan APILL merupakan salah satu solusi untuk meningkatkan keamanan dan kenyamanan kendaraan ketika berada di dalam sebuah persimpangan. Untuk memaksimalkan efisiensi sebuah sistem APILL, maka diperlukan Algoritma Greedy untuk mengatur penjadwalan lampu hijau, sehingga tidak menimbulkan kemacetan atau menyebabkan kecelakaan.**

Keywords—kendaraan, keamanan, kenyamanan, lalu lintas, APILL, efisiensi, persimpangan.

I. PENDAHULUAN

Kendaraan adalah salah satu bukti perkembangan teknologi manusia yang sangat berguna. Sejak ditemukannya mobil bermotor pada tahun 1886 oleh Karl Benz, variasi kendaraan terus meningkat. Kendaraan yang umum ditemukan sekarang adalah mobil, sepeda motor, bus, dan truk. Kendaraan yang ada sekarang memiliki ukuran yang tidak kecil, mulai dari sepeda motor yang mampu masuk ke celah – celah yang sempit, hingga truk yang membutuhkan ruang lebar agar bisa lewat.

Untuk menampung semua jenis kendaraan yang ada sekarang, salah satu solusi yang dibuat adalah jalan raya. Jalan raya memiliki banyak variasi, ada yang memiliki dua jalur, ada yang memiliki empat jalur, bahkan ada yang memiliki enam jalur. Semakin banyak jalur di sebuah jalan, maka semakin banyak kendaraan yang bisa ditampung. Untuk meningkatkan kapasitas sebuah jalan, maka bisa diberlakukan aturan *one-way* atau jalur satu arah.

Akan tetapi, apabila sebuah jalan memiliki jalur yang sedikit dan jumlah kendaraan yang banyak, maka laju kendaraan bisa tersendat. Penurunan laju kendaraan ini disebut kemacetan. Kapasitas jalan yang terlalu kecil merupakan salah satu penyebab kemacetan, tetapi masih ada beberapa penyebab lain, misalnya kecelakaan dan pembangunan di dekat jalan. Penyebab lainnya adalah persimpangan jalan, yaitu tempat dimana dua jalan atau lebih bertemu di satu titik.

Persimpangan dapat menyebabkan kemacetan karena setiap kendaraan memiliki tujuan mereka masing – masing. Di Indonesia, kendaraan yang ingin berbelok ke kiri umumnya tidak akan memotong arus lalu lintas. Akan tetapi, untuk kendaraan yang ingin berbelok ke kanan, maka kendaraan harus memotong arus lalu lintas. Pemotongan arus lalu lintas ini tentu saja akan memperlambat laju kendaraan, sehingga menimbulkan kemacetan.

Gambar 1: Persimpangan antara dua jalan yang berbeda
Sumber: https://betterprogramming.pub/javascript-type-checking-with-flow-intersection-types-8f115659f4bb

Salah satu solusi permasalahan ini adalah menggunakan Alat Pemberi Isyarat Lalu Lintas atau APILL. APILL merupakan lampu yang terpasang di sebuah persimpangan untuk mengatur arus lalu lintas sehingga tidak menyebabkan kemacetan. Semua kendaraan yang datang ke persimpangan harus mematuhi tanda yang diberikan oleh alat ini. Pelanggaran terhadap tanda yang diberikan oleh alat ini sangat berbahaya karena dapat menyebabkan kecelakaan. Oleh karena itu, APILL memerlukan sistem pengaturan waktu yang memaksimalkan efisiensi, yaitu tidak terlalu lama dalam fase lampu merah agar tidak terjadi kemacetan, dan juga tidak terlalu cepat dalam fase lampu hijau agar volume kendaraan yang lewat tidak terlalu kecil.

Sistem pengaturan waktu ini bisa dibuat dengan beberapa cara, misalnya dengan menebak durasi lampu atau menghitung rata – rata kendaraan yang lewat dan mengkalkulasi jumlah waktu yang diperlukan agar arus kendaraan tidak tersendat. Kedua cara ini akan menghasilkan durasi waktu yang konstan

dan memerlukan perubahan apabila tingkat arus lalu lintas di persimpangan tersebut meningkat. Cara lain untuk membuat sistem pengaturan waktu ini adalah dengan menggunakan *detector*. *Detector* ini akan menghitung banyaknya jumlah kendaraan yang sedang menunggu di salah satu sisi persimpangan dan mengirimkan data ke sebuah *controller* untuk menghitung durasi waktu. Cara ini akan menghasilkan durasi waktu yang dinamis sehingga sistem bisa beradaptasi terhadap fluktuasi jumlah kendaraan yang datang ke persimpangan.

Untuk menghitung durasi waktu yang dinamis, maka kita bisa menggunakan algoritma *greedy*. Perumusan masalah optimasinya adalah dengan jumlah kendaraan yang datang ke persimpangan, berapa durasi waktu yang dapat memaksimalkan jumlah kendaraan melewati persimpangan. Perumusan masalah – masalah lainnya akan dibahas di bagian selanjutnya.

II. LANDASAN TEORI

A. *Algoritma Greedy*

Di masa yang menuntut kita untuk menyelesaikan sebuah permasalahan secara tepat dan efisien, kita akan sering menemukan persoalan optimisasi. Ada dua macam permasalahan optimisasi, yaitu persoalan maksimasi dan persoalan minimasi. Algoritma *greedy* memiliki prinsip "*take what you can get now!*", sehingga algoritma ini akan mencari solusi langkah per langkah. Setiap langkah memiliki banyak pilihan yang harus dievaluasi, karena algoritma *greedy* tidak bisa mundur ke langkah sebelumnya apabila sudah membuat keputusan. Dalam setiap langkah, algoritma akan memilih optimum lokal dengan harapan bahwa langkah selanjutnya akan mengarah ke optimum global, sehingga solusi yang paling tepat bisa ditemukan.

Algoritma *greedy* memiliki elemen – elemen sebagai berikut:

- Himpunan kandidat (C)

  Himpunan ini berisi kandidat pilihan yang dapat dipilih pada setiap langkah. Algoritma akan mengambil salah satu kandidat ini dan lanjut ke langkah berikutnya. Contoh dari himpunan ini adalah kumpulan benda atau karakter.

- Himpunan solusi (S)

  Himpunan ini berisi kandidat yang sudah dipilih dari himpunan kandidat. Karena isi dari himpunan solusi berasal dari himpunan kandidat, maka himpunan solusi adalah himpunan bagian dari himpunan kandidat

- Fungsi solusi

  Fungsi ini akan menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi atau belum.

- Fungsi seleksi

  Fungsi ini akan memilih kandidat berdasarkan strategi *greedy* tertentu. Fungsi ini akan mencari kandidat dari himpunan kandidat yang paling memungkinan untuk mencapai solusi optimal. Karena algoritma *greedy* tidak dapat kembali ke langkah sebelumnya, maka pilihan yang sudah diambil tidak dapat dipertimbangkan lagi di langkah – langkah selanjutnya.

- Fungsi kelayakan

  Fungsi ini akan memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi. Kandidat yang terdapat dalam himpunan kandidat belum tentu layak untuk dipilih, misalnya apabila memasukkan kandidat tersebut ke himpunan solusi, maka solusi menjadi berlebih.

- Fungsi obyektif

  Fungsi ini akan memaksimumkan atau meminimasikan nilai solusi sesuai dengan kebutuhan.

Dengan menggunakan elemen – elemen algoritma *greedy* yang sudah dijelaskan diatas, kita dapat menyimpulkan bahwa algoritma *greedy* melibatkan pencarian sebuah himpunan bagian S (himpunan solusi) dari himpunan kandidat C, yang dalam hal ini, himpunan S harus memenuhi kriteria yang ditentukan, yaitu S memberikan suatu solusi dan S dioptimasi oleh fungsi obyektif.

Skema umum dari algoritma *greedy* terdiri dari beberapa tahapan berikut ini:

1. Inisialisasikan sebuah himpunan kosong S
2. Pilih sebuah kandidat dari himpunan kandidat C. Pemilihan harus melewati fungsi seleksi terlebih dahulu untuk menemukan kandidat yang paling optimal.
3. Buang kandidat yang sudah diambil dari himpunan kandidat C
4. Cek kelayakan kandidat yang diambil menggunakan fungsi kelayakan, dan apabila fungsi kelayakan sukses, maka kandidat tersebut bisa dimasukkan ke dalam himpunan S
5. Ulangi langkah 2 sampai 5 hingga fungsi solusi S sukses atau isi himpunan kandidat C kosong
6. Apabila fungsi solusi tercapai, maka algoritma *greedy* menemukan solusi yang optimal, tetapi apabila tidak tercapai, maka solusi tidak dapat ditemukan.

Solusi yang ditemukan oleh algoritma *greedy* sudah pasti optimum global, tetapi solusi ini belum tentu merupakan solusi optimum atau solusi yang terbaik. Hal ini disebabkan oleh dua factor, yaitu:

- Algoritma *greedy* tidak mengecek seluruh kemungkinan yang mungkin ada (seperti yang dilakukan oleh algoritma *exhaustive search*), sehingga akan ada kombinasi kandidat yang tidak dicek oleh algoritma *greedy*.
- Fungsi seleksi yang tidak tepat akan mengakibatkan algoritma *greedy* tidak menemukan solusi optimal.

Apabila solusi terbaik tidak terlalu dibutuhkan, maka algoritma *greedy* bisa digunakan untuk menghasilkan solusi hampiran (aproksimasi). Hal ini tentu akan meningkatkan

efisiensi, karena apabila algoritma *greedy* tidak dapat menemukan solusi optimal yang mutlak, maka solusi harus dicari menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menemukan solusi optimal yang mutlak.

B. Alat Pemberi Isyarat Lalu Lintas

Alat Pemberi Isyarat Lalu Lintas atau APILL adalah sebuah alat yang bertujuan untuk mengatur arus lalu lintas di dalam sebuah persimpangan. Secara umum, sebuah APILL memiliki tiga warna yang berbeda, yaitu merah, kuning, dan hijau. Warna merah memberitahu pengendara untuk berhenti, warna kuning memberitahu pengendara untuk mulai berhenti karena lampu akan berubah ke warna merah, dan warna hijau memberitahu pengendara untuk mulai berjalan.

Durasi lampu warna kuning dalam sebuah APILL biasanya sangat singkat, berada dalam jangkauan tiga sampai lima detik. Akan tetapi, untuk warna merah dan hijau, durasi yang digunakan bisa berbeda – beda antar persimpangan. Untuk warna merah yang berdurasi lama, maka jumlah kendaraan yang menumpuk di salah satu sisi persimpangan akan semakin bertambah, sehingga menimbulkan kemacetan, sebaliknya apabila warna merah berdurasi singkat, maka laju kendaraan tidak akan terlalu terhambat ketika melewati persimpangan. Di sisi lain, untuk durasi warna hijau yang lama, maka volume kendaraan yang dapat melewati persimpangan meningkat, sebaliknya untuk durasi warna hijau yang singkat, maka volume kendaraan yang dapat melewati persimpangan berkurang, sehingga mampu menimbulkan kemacetan.

Secara umum, APILL di salah satu sisi dalam persimpangan akan menunjukkan warna hijau dan sisi lainnya menunjukkan warna merah. Akan tetapi, ada persimpangan yang menunjukkan warna hijau di dua sisi persimpangan. Misalnya salah satu sisi yang menunjukkan lampu hijau adalah sisi A. Sisi lain yang menunjukkan warna hijau adalah sisi di depan sisi A. Ada juga persimpangan dengan lampu berbeda untuk kendaraan yang ingin berbelok. Hal ini bertujuan agar kendaraan yang ingin melaju lurus dan berbelok tidak saling menghambat. Untuk makalah ini, penulis akan menggunakan tipe persimpangan yang hanya memiliki satu lampu hijau dan sisi lainnya menunjukkan lampu merah.

Dalam mencari durasi waktu lampu merah dan hijau yang efisien untuk sistem APILL di sebuah persimpangan, maka perhitungan harus mengikutsertakan durasi setiap sisi persimpangan. Setiap sisi persimpangan akan memiliki durasi lampu merah dan hijau masing – masing, sehingga sistem APILL tidak boleh memiliki dua sisi persimpangan yang saling berpotongan menunjukkan warna hijau secara bersamaan. Hal ini akan mengakibatkan penumpukan kendaraan di dalam persimpangan atau bahkan kecelakaan. Untuk menghindari permasalahan ini, maka APILL di setiap sisi persimpangan harus menghitung durasi lampu merah dengan menjumlahkan durasi lampu hijau di sisi persimpangan lainnya.

Akan tetapi, durasi waktu lampu hijau berbeda dengan lampu merah. Lampu hijau tidak terikat dengan durasi lampu merah, sehingga durasi waktu yang diberikan dibebaskan. Durasi waktu lampu hijau ini yang menjadi kunci efisiensi sebuah sistem APILL dalam persimpangan. Durasi lampu hijau yang lama akan meningkatkan volume kendaraan yang masuk ke persimpangan di salah satu sisi, tetapi menimbulkan penumpukan kendaraan di sisi lain persimpangan. Untuk membentuk sistem APILL yang efisien, perlu ditemukan keseimbangan antara durasi lampu merah dan hijau, sehingga mendapatkan arus lalu lintas yang optimal.

Untuk menemukan keseimbangan durasi ini, maka ada beberapa cara yang dapat dilakukan. Cara pertama adalah menggunakan kalkulasi rata – rata banyaknya kendaraan yang biasa melewati persimpangan itu. Apabila persimpangannya tergolong kecil (artinya tidak memiliki kapasitas kendaraan yang besar), maka durasi waktu lampu hijau dan merahnya tidak perlu terlalu lama, karena jumlah kendaraan yang melewatinya juga kecil. Akan tetapi, apabila persimpangannya besar, maka volume kendaraan yang harus lewat pun meningkat, sehingga durasi waktu lampu hijau harus cukup lama dan durasi lampu merah sebaiknya singkat.

Cara kedua adalah menggunakan *detector* seperti yang sudah disebutkan. *Detector* ini akan menghitung jumlah kendaraan yang sedang menunggu di salah satu sisi persimpangan dan mengirimkan data ini ke *controller* dari sistem. *Controller* kemudian akan mencari sisi persimpangan yang membutuhkan prioritas tertinggi, artinya sisi yang memiliki jumlah kendaraan terbanyak. Sisi dengan prioritas tertinggi akan diberikan durasi lampu hijau yang lebih lama, agar volume kendaraan yang lewat bisa mengurangi penumpukan. *Detector* juga akan menghitung jumlah kendaraan setelah lampu menjadi hijau, dan apabila jumlah kendaraan sudah berkurang signifikan, maka data ini akan dikirim ke *controller* dan lampu di sisi itu diubah menjadi merah.

Siklus kerja *detector* dan *controller* akan terus berlangsung dan tidak memerlukan perubahan oleh manusia, karena sistem akan beradaptasi terhadap volume kendaraan yang datang ke persimpangan. Cara ini akan meningkatkan efisiensi dari sebuah persimpangan, karena apabila menggunakan cara kalkulasi yang pertama, maka durasi yang ditetapkan konstan, sehingga tidak dapat memprediksi apabila volume kendaraan tiba – tiba bertambah. Dengan tren jumlah kendaraan yang semakin meningkat, maka cara kedua menjadi pilihan yang paling optimal untuk menyelesaikan permasalahan penjadwalan durasi APILL di sebuah persimpangan.

III. HASIL DAN PEMBAHASAN

A. Asumsi dan Batasan Algoritma

Persoalan penjadwalan durasi waktu APILL ini mirip dengan *integer knapsack problem*. *Integer knapsack problem* akan mencari cara untuk memilih objek yang akan dimasukkan ke dalam *knapsack* sedemikian sehingga total keuntungan yang didapatkan maksimal. Dalam persoalan penjadwalan durasi waktu APILL, algoritma akan memilih sisi persimpangan mana yang akan diberikan lampu hijau sedemikian sehingga volume kendaraan yang melewati persimpangan maksimal. Algoritma *greedy* akan menerima informasi banyaknya kendaraan yang sedang menunggu di setiap sisi persimpangan.

Seperti yang telah disinggung di bagian sebelumnya, makalah ini akan membahas penggunaan algoritma *greedy*

untuk penjadwalan durasi APILL pada persimpangan yang hanya memperbolehkan satu sisi berjalan saja. Artinya tidak boleh ada dua sisi atau lebih dari persimpangan yang menunjukkan lampu hijau. Urutan dari lampu hijau antar sisi juga harus teratur, artinya apabila sisi utara menunjukkan warna hijau, maka yang selanjutnya dapat menunjukkan warna hijau adalah sisi timur, kemudian sisi selatan, dan berakhir di sisi barat. Beberapa asumsi yang digunakan di dalam makalah ini antara lain:

- Asumsi pertama yang digunakan adalah kecepatan kendaraan. Setiap jenis kendaraan tentu saja akan memiliki kecepatan yang berbeda – beda. Sepeda motor akan lebih cepat berjalan dibandingkan dengan sebuah truk. Algoritma akan mengasumsikan bahwa kecepatan setiap kendaraan sama, sehingga tidak ada penumpukan kendaraan akibat kecepatan yang berbeda – beda.
- Asumsi selanjutnya yang akan digunakan adalah banyaknya sisi dalam persimpangan. Secara umum, persimpangan terdiri atas empat sisi. Akan tetapi terdapat persimpangan yang memiliki tiga sisi, lima sisi, atau bahkan lebih. Algoritma akan mengasumsikan bahwa di dalam sebuah persimpangan terdapat empat sisi.
- Asumsi terakhir yang akan digunakan adalah respon manusia. Setiap pengendara akan memiliki reflek yang berbeda terhadap sinyal lampu APILL. Algoritma akan mengasumsikan bahwa respon setiap pengendara sama dan langsung berjalan setelah lampu menjadi hijau.
- Asumsi untuk kecepatan arus lalu lintas adalah lima kendaraan per detik yang dapat masuk ke persimpangan. Durasi standar untuk lampu hijau adalah 4 detik. Nilai ini bisa bertambah atau berkurang tergantung algoritma. Asumsi durasi standar ini tidak merefleksikan durasi APILL di kehidupan nyata, tetapi digunakan untuk mempermudah perhitungan.

Data yang akan disediakan untuk algoritma ini terdiri dari simbol sisi persimpangan dan jumlah kendaraan yang sedang menunggu. Simbol sisi persimpangan akan mengikuti arah jarum jam, dengan huruf A berada di sisi utara. Jumlah kendaraan tidak melihat jenis kendaraan, sehingga jumlahnya hanya ada satu. Data yang akan digunakan dalam makalah ini adalah sebagai berikut:

- Siklus pertama

Sisi persimpangan	Jumlah kendaraan yang datang
A	31
B	42
C	23
D	22

- Siklus kedua

Sisi persimpangan	Jumlah kendaraan yang datang
A	75
B	26
C	30
D	43

- Siklus ketiga

Sisi persimpangan	Jumlah kendaraan yang datang
A	49
B	52
C	57
D	10

Elemen – elemen algoritma *greedy* dalam persoalan penjadwalan durasi APILL antara lain:

- Himpunan kandidat

  Himpunan kandidat untuk algoritma *greedy* berisi tabel waktu lampu hijau menurut banyaknya kendaraan yang menunggu

Banyaknya kendaraan yang menunggu	Durasi waktu lampu hijau
>=20 dan < 30	4
>=30 dan < 40	6
>=40 dan < 50	8
>=50 dan < 60	10
>=60	12

- Himpunan solusi

  Himpunan solusi untuk penjadwalan durasi APILL berisi durasi lampu hijau setiap sisi persimpangan.

- Fungsi Obyektif

  Algoritma harus memaksimalkan volume kendaraan yang melewati persimpangan.

- Fungsi Solusi

  Solusi ditemukan ketika semua sisi persimpangan sudah mendapat giliran menunjukkan lampu hijau.

- Fungsi Kelayakan

  Kandidat yang cocok menjadi solusi adalah sisi yang belum pernah menunjukkan lampu hijau selama satu siklus. Urutan lampu hijau harus mengurut searah jarum jam, sehingga urutan dimulai dari utara, timur, selatan, dan berakhir di barat.

- Fungsi Seleksi

  Kandidat durasi yang dipilih harus mengambil salah satu durasi dari himpunan kandidat sesuai dengan jumlah kendaraan yang sedang menunggu di sisi persimpangan. Apabila jumlah kendaraan yang menunggu kurang dari 20, maka durasi waktu lampu hijau yang diperbolehkan adalah

$$(jumlah\ kendaraan\ yang\ menunggu)\ div\ 5 \quad (1)$$

Hal ini bertujuan agar tidak ada durasi yang terbuang sia – sia dan mengurangi durasi lampu merah di sisi persimpangan lainnya. Apabila jumlah kendaraan yang menunggu lebih atau sama dengan 70, maka durasi waktu lampu hijau yang harus ditambahkan adalah

$$(jumlah\ kendaraan\ yang\ menunggu - 70)\ div\ 5 \quad (2)$$

Hasil perhitungan durasi waktu diatas harus ditambahkan ke durasi untuk jumlah kendaraan diatas 60 kendaraan.

B. Implementasi Algoritma Greedy

1) Siklus pertama

- Sisi persimpangan A

  Jumlah kendaraan yang menunggu = 31 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 6 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 30 kendaraan

  Himpunan solusi S : {6}

  Sisa kendaraan di sisi persimpangan A = 1 kendaraan

- Sisi persimpangan B

  Jumlah kendaraan yang menunggu = 42 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 8 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 30 + 40 = 70 kendaraan

  Himpunan solusi S : {6, 8}

  Sisa kendaraan di sisi persimpangan B = 2 kendaraan

- Sisi persimpangan C

  Jumlah kendaraan yang menunggu = 23 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 4 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 30 + 40 + 20 = 90 kendaraan

  Himpunan solusi S : {6, 8, 4}

  Sisa kendaraan di sisi persimpangan C = 3 kendaraan

- Sisi persimpangan D

  Jumlah kendaraan yang menunggu = 22 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 4 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 30 + 40 + 20 + 20 = 110 kendaraan

  Himpunan solusi S : {6, 8, 4, 4}

  Sisa kendaraan di sisi persimpangan D = 2 kendaraan

  Fungsi solusi berhasil karena semua sisi persimpangan sudah mendapat giliran menunjukkan lampu hijau.

2) Siklus kedua

- Sisi persimpangan A

  Jumlah kendaraan yang baru datang = 75 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 1 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 76 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 12 detik

  Karena jumlah kendaraan melebihi 70, maka menggunakan persamaan (2), didapatkan durasi waktu yang perlu ditambahkan = 1 detik, sehingga total durasi waktu lampu hijau adalah 13 detik.

  Volume kendaraan yang lewat ke dalam persimpangan = 65 kendaraan

  Himpunan solusi S : {13}

  Sisa kendaraan di sisi persimpangan A = 11 kendaraan

- Sisi persimpangan B

  Jumlah kendaraan yang baru datang = 26 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 2 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 28 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 4 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 65 + 20 = 85 kendaraan

  Himpunan solusi S : {13, 4}

  Sisa kendaraan di sisi persimpangan B = 8 kendaraan

- Sisi persimpangan C

  Jumlah kendaraan yang baru datang = 30 kendaraan

Jumlah kendaraan yang menunggu sebelumnya = 3 kendaraan

Jumlah kendaraan yang menunggu sekarang = 33 kendaraan

Fungsi seleksi mengambil durasi waktu lampu hijau = 6 detik

Volume kendaraan yang lewat ke dalam persimpangan = 65 + 20 + 30 = 115 kendaraan

Himpunan solusi S : {13, 4, 6}

Sisa kendaraan di sisi persimpangan C = 3 kendaraan

- Sisi persimpangan D

  Jumlah kendaraan yang baru datang = 43 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 2 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 45 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 8 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 65 + 20 + 30 + 40 = 155 kendaraan

  Himpunan solusi S : {13, 4, 6, 8}

  Sisa kendaraan di sisi persimpangan D = 5 kendaraan

  Fungsi solusi berhasil karena semua sisi persimpangan sudah mendapat giliran menunjukkan lampu hijau.

3) Siklus ketiga

- Sisi persimpangan A

  Jumlah kendaraan yang baru datang = 49 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 11 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 60 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 12 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 60 kendaraan

  Himpunan solusi S : {12}

  Sisa kendaraan di sisi persimpangan A = 0 kendaraan

- Sisi persimpangan B

  Jumlah kendaraan yang baru datang = 52 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 8 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 60 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 12 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 60 + 60 = 120 kendaraan

  Himpunan solusi S : {12, 12}

  Sisa kendaraan di sisi persimpangan B = 0 kendaraan

- Sisi persimpangan C

  Jumlah kendaraan yang baru datang = 47 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 3 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 50 kendaraan

  Fungsi seleksi mengambil durasi waktu lampu hijau = 10 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 60 + 60 + 50 = 170 kendaraan

  Himpunan solusi S : {12, 12, 10}

  Sisa kendaraan di sisi persimpangan C = 0 kendaraan

- Sisi persimpangan D

  Jumlah kendaraan yang baru datang = 10 kendaraan

  Jumlah kendaraan yang menunggu sebelumnya = 5 kendaraan

  Jumlah kendaraan yang menunggu sekarang = 15 kendaraan

  Karena jumlah kendaraan kurang dari 20, maka menggunakan persamaan (1), didapatkan durasi waktu yang harus digunakan = 3 detik

  Volume kendaraan yang lewat ke dalam persimpangan = 60 + 60 + 50 + 15 = 185 kendaraan

  Himpunan solusi S : {12, 12, 10, 3}

  Sisa kendaraan di sisi persimpangan D = 0 kendaraan

Fungsi solusi berhasil karena semua sisi persimpangan sudah mendapat giliran menunjukkan lampu hijau.

Hasil implementasi algoritma diatas hanya mencakup tiga siklus sistem APILL untuk menyederhanakan permasalahan. Pada sistem APILL yang sesungguhnya, algoritma ini diharapkan dapat berjalan terus – menerus, beradaptasi terhadap perubahan volume kendaraan yang datang, dan menghasilkan durasi lampu hijau yang tepat. Hasil perhitungan rata – rata waktu menunggu setiap siklus menunjukkan 16.5 detik untuk siklus pertama, 23.25 detik untuk siklus kedua, dan 27.75 detik untuk siklus ketiga. Perhitungan volume kendaraan yang datang ke persimpangan adalah 118 kendaraan untuk siklus pertama, 174 kendaraan untuk siklus kedua, dan 168 kendaraan untuk siklus ketiga. Ini menunjukkan bahwa algoritma *greedy* dengan fungsi seleksi diatas tidak akan selalu menghasilkan solusi yang optimal.

IV. Kesimpulan

Algoritma *greedy* dapat digunakan untuk menyelesaikan permasalahan kemacetan yang sering terjadi di persimpangan dalam kota. Akan tetapi, untuk mendapatkan hasil yang paling optimal dari algoritma *greedy* ini, maka perlu dibentuk sebuah fungsi seleksi yang optimal, karena fungsi ini menjadi kunci pemiihan kandidat dari himpunan kandidat. Fungsi seleksi yang kurang optimal akan menghasilkan solusi yang tidak efisien, sehingga kurang efektif untuk mengurangi kemacetan.

Contoh penggunaan yang diberikan di dalam makalah ini juga sangat menyederhanakan persoalan yang terjadi di dunia nyata. Seperti yang disebutkan di bagian sebelumnya, sistem APILL tidak selalu mengharuskan satu sisi untuk menunjukkan lampu hijau. Ada persimpangan yang memiliki lampu berbeda untuk tujuan berbelok dan ada juga persimpangan yang memperbolehkan dua sisi persimpangan menunjukkan lampu hijau asalkan sisinya saling berhadapan. Apabila algoritma *greedy* ingin diimplementasikan untuk persimpangan dengan lampu yang berbeda untuk berbelok, maka diperlukan data jumlah kendaraan yang ingin berjalan lurus dan berbelok, sehingga algoritma dapat mengkalkulasi durasi waktu dengan data yang sesuai.

Penggunaan algoritma *greedy* ini jauh lebih baik dibandingkan strategi mengkalkulasi rata – rata kendaraan yang lewat secara manual, karena hasil perhitungan tidak dinamis dan harus terus – menerus dipantau. Sedangkan penggunaan algoritma *greedy* tidak akan memerlukan pemeliharaan secara ekstensif karena hasil perhitungan akan beradaptasi sesuai dengan volume kendaraan. Algoritma *greedy* juga memiliki efisiensi waktu yang lebih baik jika dibandingkan dengan algoritma *brute force*. Algoritma *greedy* tidak memperhitungkan seluruh kemungkinan permutasi durasi waktu, tetapi hanya menghitung pilihan yang sesuai dengan fungsi seleksi, mengurangi banyaknya permutasi yang harus dihitung.

Video Link at Youtube

Video demonstrasi isi makalah dapat dilihat melalui tautan berikut: bit.ly/VideoDemoMakalahAPILL

Acknowledgment

Segala puji syukur penulis panjatkan bagi Tuhan Yang Maha Esa karena telah memberikan penulis kelancaran dalam menulis makalah yang berjudul "Penerapan Algoritma Greedy dalam Membantu Penjadwalan Alat Pemberi Isyarat Lalu Lintas". Penulis juga ingin menyampaikan terima kasih kepada orang tua penulis atas dukungan yang diberikan kepada penulis selama proses pembuatan makalah. Penulis ingin menyampaikan rasa terima kasih sebesar – besarnya kepada Ibu Nur Ulfa Maulidevi selaku dosen Strategi Algoritma K02 yang telah membimbing penulis selama proses perkuliahan. Penulis menyadari bahwa makalah ini jauh dari kata sempurna. Oleh karena itu, penulis berharap bahwa makalah ini dapat digunakan semaksimal mungkin dan dikembangkan lebih lagi agar memberi dampak yang besar ke masyarakat luas.

References

[1] R. Munir. "Algoritma Greedy (Bagian 1)", diakses pada 21 Mei 2022 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

[2] Anonim. "Begini Sejarah Awal dari Lampu Lalu Lintas", diakses pada 21 Mei 2022 dari https://www.suzuki.co.id/tips-trik/begini-sejarah-awal-dari-lampu-lalu-lintas

[3] C. Lauren. "Who invented the car?", diakses pada 21 Mei 2022 dari https://www.livescience.com/37538-who-invented-the-car.html

[4] M. Tom. "Vehicle actuated signals", diakses pada 21 Mei 2022 dari https://www.civil.iitb.ac.in/tvm/nptel/576_VAS/web/web.html

[5] Anonim. "Why Does Traffic Happen?", diakses pada 21 Mei 2022 dari https://www.wawanesa.com/us/blog/why-does-traffic-happen

[6] S. Puji, M. Wisnu, dan H. Dian. "Reducing a congestion with introduce the greedy algorithm on traffic light control", diakses pada 21 Mei 2022 dari https://iopscience.iop.org/article/10.1088/1742-6596/974/1/012013/pdf

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, 21 Mei 2022

Yakobus Iryanto Prasethio 13520104