Jurnal Bina Komputer
JBK, Vol. 2, No. 2, Juli 2020: 89-98
Received: 5 Mei 2020; Revised: 18 Mei 2020; Accepted: 15 Juli 2020

MODEL SIMULASI PENYELESAIAN MASALAH PERJALANAN PENJUAL MENGGUNAKAN PENDEKATAN KECERDASAN BUATAN, OPTIMISASI KOLONI SEMUT

Misinem Misinem[1], Tri Basuki Kurniawan[2,*], Astried Astried[3], Joan Angelina Widians[4]

[1]Program Studi Teknik Komputer, Universitas Bina Darma
[2]Program Studi Sistem Informasi, Universitas Bina Darma
[3]Program Studi Sistem Informasi, Universitas Riau
[4]Program Studi Teknik Informatika, Universitas Mulawarman
email :{[1]misinem, [2]tribasukikurniawan}@binadarma.ac.id,
[3]astried@lecturer.unri.ac.id, [4]angel.unmul@gmail.com

Abstract

One of the interesting things in the software development area is the discovery of optimization algorithms. A lot of complicated work that needs to be done manually will require enormous time and effort. With the existence of a complicated job, the optimization algorithms previously can be done more comfortably and faster. They are even giving support to get the best solution. In this research, a software simulation model will be created to solve the traveling salesman problem by using an ant colony optimization to provide visuals to users of issues that can be done step by step. The development program was followed the Extreme Programming (XP) software development method on the Windows operating system using the C # programming language in Visual Studio 2019. The results of the study found the program can provide comfort and clear visualizations/simulations that can be used by users.
Keywords: *Travelling Salesman Problem, TSP, Ant Colony Optimization, ACO, Simulation model*

Abstrak

Salah satu hal yang menarik dalam bidang perangkat lunak adalah ditemukannya algoritma pengoptimisasian. Banyak pekerjaan yang rumit dan kompleks yang akan mustahil untuk dilakukan secara manual ataupun kalau terpaksa dilakukan dengan cara manual akan membutuhkan waktu dan tenaga yang sangat besar. Dengan adanya algoritma optimisasian pekerjaan yang rumit dan kompleks tadi dapat dilakukan dengan lebih mudah dan lebih cepat. Bahkan juga memeberikan jaminan secara teoritis, untuk mendapatkan solusi yang terbaik. Dalam penelitian ini akan dibangun sebuah model simulasi perangkat lunak untuk menyelesaikan masalah

perjalanan penjual dengan menggunakan algoritma optimisasi koloni semut, untuk memberikan visual bagi pengguna bagaimana masalah tersebut dapat diselesaikan secara Langkah demi Langkah. Pembangunan program simulasi menggunakan metode pengembangan perangkat lunak *Extreme Programming* (XP) pada lingkungan system operasi Windows dengan menggunakan Bahasa pemrograman C# pada Visual Studio 2019. Hasil dari penelitian didapati, program dapat memberikan visualisasi/simulasi yang baik kepada pengguna.
Kata kunci: *Masalah Perjalanan Penjual, TSP, Optmisasi Koloni Semut, ACO, Model Simulasi*

1. PENDAHULUAN

Bidang komputer, merupakan bidang teknik yang paling rumit dan komplek yang pernah dipelajari atau ditemukan oleh manusia dan juga merupakan bidang yang paling cepat perkembangannya. Hampir semua bidang, pada masa ini, tidak ada yang tidak ikut memanfaatkan kemampuan ajaib dari komputer.

Komputer telah mewarnai dan menguasai kegiatan hidup kita sehari-hari. Baik itu komputer dalam bentuk komputer pribadi (pc, *personal computer*) dan *laptop* maupun dalam bentuk telepon pintar (*smart phone*). Bagi manusia modern, hampir setiap hari kita akan berhadapan dengan komputer maupun peralatan-peralatan cerdas lainnya. Bahkan kehidupan kita menjadi sangat tergantung dengan peralatan peralatan tersebut.

Kalau kita amati lagi, yang pintar sebenarnya bukan peralatannya, tetapi terletak pada perangkat lunak (*software*) yang mengatur dan menggerakan peralatan tersebut. Perangkat lunak tersebut disusun oleh seorang atau sekelompok pemrogram dengan menggunakan bahasa pemrograman tertentu dengan metode tertentu, sehingga dapat mengatur dan memberikan kemampuan kepada peralatan tersebut untuk dapat membantu pekerjaan kita sehari-hari.

Kebutuhan manusia modern saat ini sudah sangat beragam dan kompleks, sehingga harapan kepada kemampuan peralatan atau perangkat lunak juga telah jauh meningkat. Dahulu, peralatan dengan kemampuan sederhana saja, sudah mencukupi kebutuhan kita. Tetapi dengan tingkat beban pekerjaan dan kegiatan yang juga meningkat, kemampuan peralatan juga diharap dapat lebih meningkat, sehingga kebutuhan dan ketersediaan peralatan dan perangkat lunak yang mempunyai kemampuan yang baik sangat diperlukan.

Salah satu hal yang menarik dalam bidang perangkat lunak adalah ditemukannya algoritma pengoptimisasian. Banyak pekerjaan yang rumit dan kompleks yang akan mustahil untuk dilakukan secara manual ataupun kalau terpaksa dilakukan dengan cara manual akan membutuhkan waktu dan tenaga yang sangat besar. Selain itu, sering kali tidak ada jaminan untuk mendapatkan solusi yang baik ataupun yang terbaik (Ilwaru et al. 2017). Dengan adanya algoritma optimisasian pekerjaan yang rumit dan kompleks tadi dapat dilakukan dengan lebih mudah dan lebih cepat. Bahkan juga memeberikan jaminan secara teoritis, untuk mendapatkan solusi yang terbaik.

Algoritma pengoptimisasian ini dapat diterapkan hampir di semua bidang ilmu, antara lain bidang teknik, sains, ilmu sosial, ekonomi dan bisnis. Banyak permasalahan dibidang teknik, sains dan ekonomi yang dapat dinyatakan sebagai permasalahan optimisasi seperti meminimalkan biaya, waktu dan resiko atau memaksimalkan keuntungan dan kualitas (Anis, Nandiroh, and Utami 2007).

Salah satu masalah yang dapat diselesaikan menggunakan metode optimisasi adalah masalah perjalanan penjual, atau dalam Bahasa Inggeris dikenal dengan Travelling Salesman Problem (TSP). TSP adalah permasalahan yang sudah cukup dikenal di dunia optimisasi (benchmark problem). Pokok permasalahan dari TSP adalah seorang penjual (saleman) harus mengunjungi sejumlah kota yang diketahui jarak antara satu kota ke kota yang lainnya. Semua kota yang ada harus dikunjungi oleh penjual tersebut dan kota tersebut hanya boleh dikunjungi tepat satu kali.

Permasalahannya adalah bagaimana penjual tersebut dapat mengatur rute perjalanannya sehingga jarak yang ditempuhnya merupakan rute yang optimum yaitu jarak minimum terbaik (jarak terpendek yang akan ditempuh oleh penjual tersebut).

Dalam penelitian ini, akan dibahas satu model simulasi untuk menyelesaikan permasalahan TSP, menggunakan pendekatan kecerdasan buatan, dalam hal ini metode optimisasi koloni semut, atau lebih dikenal dengan Ant Colony Optimization (ACO) (T.B. Kurniawan et al. 2009).

2. METODOLOGI PENELITAN

2.1 Metode Pengembangan Perangkat Lunak

Metode pengembangan perangkat lunak yang digunakan dalam penelitian ini adalah metode Extreme Programming (XP), yaitu salah satu dari metode Agile. Dasar dari metode XP adalah menulis kode program dan lalu melakukan pengujian (Alshayeb and Li 2006). XP banyak digunakan oleh pengembang perangkat lunak karena lebih menitik-beratkan kepada kepuasan pengguna, berbanding kepada menyelesaikan dokumentasi yang mungkin tidak terlalu diperlukan secara lengkap dan menyeluruh pada saat tahap awal perangkat lunak sedang dikembangkan, dengan pertimbangan dokumentasi tersebut masih akan selalu berkembang dan berubah. XP memberikan keleluasaan kepada pelanggan untuk dapat mengubah keperluan mereka walaupun pada tahapan akhir pengembangan perangkat lunak.

Tugas utama yang dilakukan dalam XP adalah (Fojtik 2011):

- Perencanaan dan Mengaturan
- Desain
- Menyusun kode program
- Pengujian

2.2 Travelling Salesman Problem (TSP)

Permasalahan matematik yang berkaitan dengan TSP mulai muncul sekitar tahun 1800-an. Masalah ini dikemukakan oleh dua orang matematikawan, yaitu Sir William Rowan Hamilton yang berasal dari Irlandia dan Thomas Penyngton Kirkman yang berasal dari Inggris (Khan and Maiti 2019). Bentuk umum dari persoalan TSP pertama kali dipelajari oleh para matematikawan mulai tahun1930-an oleh Karl Menger di Vienna dan Harvard (Candrawati and Kadyanan 2017). Persoalan tersebut kemudian dikembangkan oleh Hassler Whitney dan Merril Flood di Princeton.

TSP secara umum merupakan masalah optimisasi yang kompleks, di mana dengan bertambahnya variabel secara linear, maka waktu yang dibutuhkan akan meningkat secara eksponensial. Mengikut jejak dari satu kota ke kota yang lainnya, maka TSP dapat dibedakan ke dalam dua jenis, yaitu TSP yang symmetric, jika jejak dari kota A ke kota B adalah sama dengan jarak dari kota B ke kota A. Sedangkan TSP yang *asymmetric*, jika sebaliknya. Untuk *asymmetric*, jumlah kombinasi perjalanan (solusi) yang mungkin dihasilkan adalah $(n-1)!$. Sedangkan untuk yang *symmetric*, jumlah solusi yang mungkin adalah setengah dari jumlah solusi yang *asymmetric*.

Jumlah solusi untuk n kota pada kasus TSP yang symmetric adalah $(n-1)!/2$. Seandainya kita mempunyai 10 buah kota (n), maka akan ada 181.440 kemungkinan solusi dan untuk 15 buah kota, akan ada 43.589.145.600 (43 milyar lebih) kemungkinan solusi. Tidak mungkin bagi kita untuk memeriksa dan menghitung satu per satu solusi tersebut dan memilih satu solusi yang paling baik. Selain diperlukan waktu yang sangat lama, juga memerlukan usaha yang sangat banyak. Artinya tidak efesien dan efektif, jika dilakukan pencarian solusi terbaik secara manual. Untuk itulah, diperlukan adanya algoritma optimisasi. Masalah seperti ini disebut juga masalah *non-deterministic polynomial-time hardness* (Fortnow 1997) atau disingkat dengan *np-hard problem*.

2.3 Proses Optimisasi

Optimisasi ialah suatu proses untuk mencapai hasil yang ideal atau optimal (nilai efektif yang dapat dicapai). Dalam disiplin ilmu matematika, optimisasi merujuk pada studi permasalah yang mencoba untuk mencari nilai minimal atau maksimal dari suatu fungsi riil. Untuk dapat mecapai nilai optimal, baik minimal ataupun maksimal tersebut, secara sistematis dilakukan pemilihan nilai variable yang akan memberikan solusi optimal (terbaik) (Anis, Nandiroh, and Utami 2007).

Konsep optimisasi sudah dipakai sejak jaman prasejarah. Hal ini dapat dibuktikan dengan adanya saluran-saluran air yang ditemukan disitus-situs prasejarah. Saluran-saluran air ini dipakai untuk mengoptimalkan pengunaan air. Hal ini mengindikasikan bahwa konsep optimisasi merupakan bagian dari kehidupan manusia sejak lama. Permasalah pengaturan air masih dijumpai dalam masyarakat masa kini, hanya saja penyelesaiannya sudah menggunakan metode optimisasi yang modern.

Algoritma optimisasi adalah metode pencarian, yang bertujuan mencari sebuah solusi bagi sebuah masalah optimisasi, yang memberikan nilai optimal yang mungkin bergantung pada beberapa batasan. Walaupun definisi dari optimisasi sangat sederhana, tetapi dibalik itu terdapat hal yang sangat komplek yang perlu diselesaikan. Sebagai contoh, solusi yang akan dicapai mungkin terdiri dari beberapa jenis data yang berbeda, dengan batasan yang sangat ketat dan ruang pencarian (*search space*) yang sangat komplek (*convoluted*) dengan banyak calon-calon solusi, serta karakteristik masalah yang mungkin berubah sejalan dengan waktu atau kondisi tertentu (*dynamic problem*) atau kualitas dari objektif yang sedang dicari nilai optimalnya terjadi konflik antara satu objektif dengan objektif lainnya (*multi-objective problem*).

2.5 Ant Colony Optimization (ACO)

Pengamatan tentang kebiasan dan prilaku sekumpulan semut telah mengispirasi banyak algoritma berdasarkan pada kerja semut (*ant-based algorithms*), yang digunakan untuk menyelesaikan masalah optimisasi combinatorial pada ruang carian diskrit. Selain semut, banyak prilaku sederhana dari mahkluk hidup lain yang mengilhami para peneliti untuk menemukan metode dan algoritma baru dalam bidang optimisasi maupun bidang kecerdasan buatan (*Artificial Intelligence*, *AI*) lainnya.

Bagaimana sekumpulan semut dapat menemukan jalan terpendek antara sumber makanan yang mereka ditemukan dengan sarang tempat mereka tinggal tanpa adanya petunjuk? Penelitian tentang kebiasaan dan prilaku beberapa jenis semut menunjukkan adanya pencarian secara acak untuk menemukan sumber makanan. Namun, setelah sumber makanan ditemukan, aktifitas semut semakin terorganisir dengan semakin banyaknya semut yang mengikuti jejak terpendek. Lambat laun akhirnya semua semut mengikuti jejak yang sama. Terjadi aktifitas 'menularkan pengetahuan' tentang jarak terpendek dari satu semut kepada semut yang lain.

Aktifitas 'menularkan pengetahuan' ini berbeda antara satu jenis semut dengan jenis semut lainnya, baik melalui kontak langsung, maupun komunikasi secara tidak langsung. Kebanyakan jenis semut, berkomunikasi melalui jejak *pheromone*. Komunikasi tidak langsung dimana semut mengubah lingkungannya (dengan meletakkan *pheromone*) untuk mempengaruhi tingkah laku semut yang lain dikenal dengan istilah *stigmergy*.

Optimisasi Koloni Semut (*Ant Colony Optimization*, *ACO*) yang selanjutnya akan kita singkat dengan *ACO* saja, adalah satu dari banyak algoritma yang diilhami oleh prilaku mahluk hidup. Seperti Algoritma Genetik yang mengambil prinsip-prinsip teori genetika, Optimisasi Kumpulan Partikel (*Particle Swarm Optimization*) yang mengambil prilaku sekawanan hewan, seperti lebah, yang berkerumun pada sesuatu dan algoritma lainnya.

Algoritma pertama yang dirancang oleh Dorigo di dalam disertasi PhDnya, adalah *Ant System* (*AS*) yang merupakan cikal bakal dari algoritma lainnya yang berdasarkan prilaku semut. Selanjutnya, Dorigo melalukan beberapa penambahan dan perbaikan dari algoritma AS, dengan diusulkannya Algoritma *Ant Colony Optimization* (ACO). Penambahan dan perbaikan tersebut meliputi tiga proses utama yang ada pada ACO, yaitu *state transition rule*, *global updating rule* dan *local pheromone updating rule*, berupa mekanisme pemilihan jejak bagi semut, proses penghargaan (*reward*) yang akan memberikan keuntungan bagi solusi yang terbaik (dalam hal ini, konsentrasi *pheromone*) dan mekanisme pengurangan konsentrasi *pheromone* dengan adanya penguapan (*evaporation*). Untuk keterangan lebih lanjut mengenai formulasi untuk setiap rule yang digunakan dapat dilihat pada penelitian yang dilakukan Kurniawan dkk (Tri Basuki Kurniawan et al. 2008).

Secara sederhananya, algoritma ACO dapat dijelaskan seperti pada *pseudo-code* yang ada pada gambar 1, berikut ini.

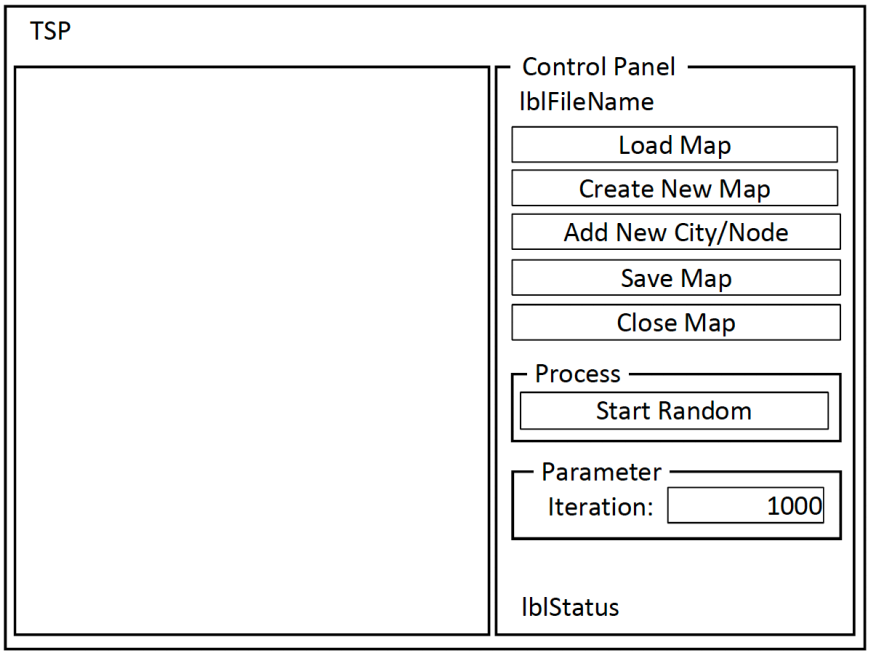
```
1:   // Proses awal (initialization process)
2:   set nilai awal untuk parameter β, ρ, q0, nk , tmax
3:   hitung_nilai τ0
4:   for setiap_jejak(i, j) do
5:      τ(i,j) = τ0 // isi matrik pheromone dengan nilai awal
6:   endfor
7:   t = 1 //set iterasi pertama
8:    repeat
9:      letakkan_semua_semut, k = 1 . . . nk
10:      // (nk adalah jumlah semut)
11:      // secara acak di kota untuk mulai perjalanan
12:      repeat
13:         for setiap_semut, k = 1 . . . nk do
14:            setiap_semut memilih kota selanjutnya
                  menggunakan transition rule
15:            aplikasikan local pheromone updating
16:            // untuk setiap perpindahan semut ke kota
                  berikutnya
17:         endfor
18:      until semua_kota_dikunjungi
19:      aplikasikan global pheromone updating
20:      // hanya untuk solusi yang terbaik saja
21:      t = t + 1
22:   until t = tmax
```

Gambar 1. Pseudo-code Algoritma ACO

2.6 Desain Model Simulasi

Pada tahap awal pembuatan program, kita terlebih dahulu perlu merancang antarmuka (*interface*) atau tampilan program kita. Hal ini akan lebih praktis, dikarenakan program yang akan kita bangun adalah program yang mempunyai tampilan grafis di Windows.

Secara garis besarnya, tampilan dari program kita nantinya seperti pada gambar 2 berikut ini.



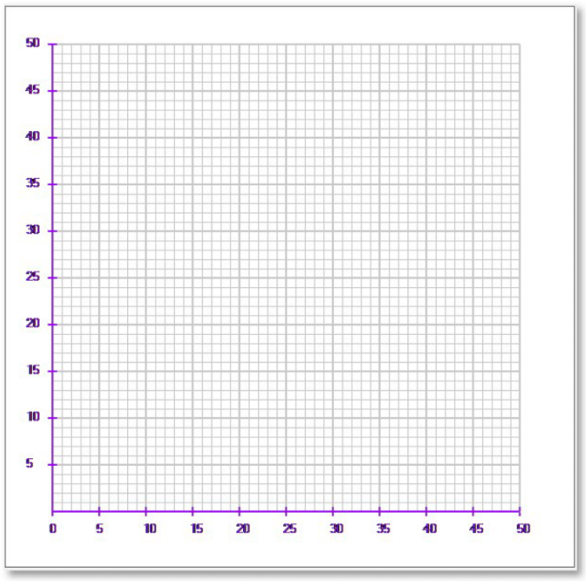
Gambar 2. Desain *mock-up* tampilan utama.

Berikutnya, desain objek yang akan digunakan dalam program ini, dapat dilihat pada gambar 3, berupa objek Position, Node dan Path.

```
1:   namespace TSP
2:   {
3:       class Position
4:      {
5:          public int x { get; set; }
6:          public int y { get; set; }
7:      }
8:      class Node
9:      {
10:          public int display { get; set; }
11:          public Position location { get; set; }
12:          public Position geo { get; set; }
13:      }
14:      class Path
15:      {
16:          public int fr { get; set; }
17:          public int to { get; set; }
18:          public double distance { get; set; }
19:      }
20:  }
```

Gambar 3. Desain objek Position, Node dan Path

Sebagai media simulasinya, dibuat sebuah tampilan untuk menampilkan simulasi dari node-node yang menggambarkan kota-kota yang akan dikunjungi oleh sang penjual. Desain media simulasi yang digunakan, dapat dilihat seperti pada gambar 4 dan gambar 5, berikut ini.



Gambar 4. Desain media simulasi

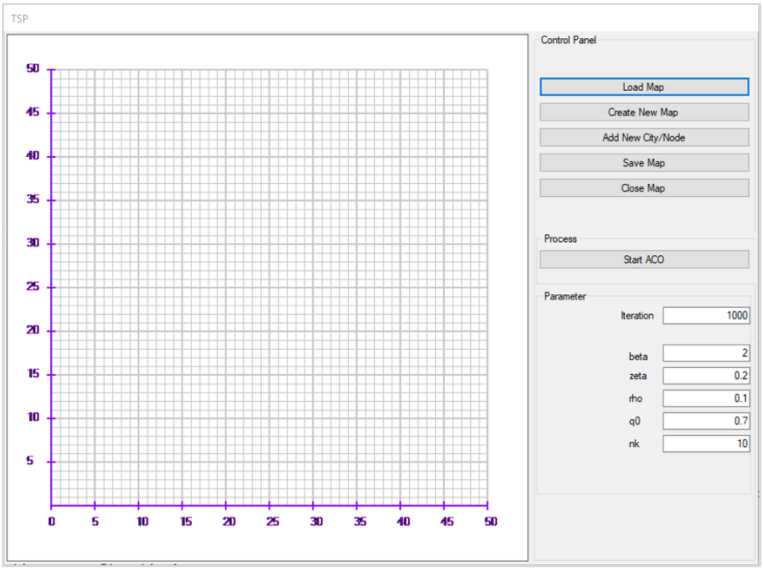
```
1:   int wid = pic.ClientSize.Width;
2:   int hgt = pic.ClientSize.Height;
3:   if (wid < 1 || hgt < 1) return;
4:   Bitmap bm = new Bitmap(wid, hgt);
5:   Graphics gr = Graphics.FromImage(bm);
6:
7:   //coloums
8:   for (int x = 50; x <= 550; x += 10)
9:   {
10:     gr.DrawLine(linePen, x, 40, x, 540);
11:  }
12:  for (int x = 50; x < 560; x += 50)
13:  {
14:     gr.DrawLine(gridPen, x, 40, x, 540);
15:     gr.DrawLine(barPen, x, 535, x, 545);
16:     gr.DrawString((((x / 10) - 5)).ToString(), new Font("Sans
           Serif", 10, FontStyle.Regular), linebrush, x - 5, 550);
17:  }
18:
19:  //rows
20:  for (int x = 40; x <= 540; x += 10)
21:  {
22:     gr.DrawLine(linePen, 50, x, 550, x);
23:  }
24:  for (int x = 40; x < 500; x += 50)
25:  {
26:     gr.DrawLine(gridPen, 50, x, 550, x);
27:     gr.DrawLine(barPen, 45, x, 55, x);
28:     gr.DrawString((55 - ((x / 10) + 1)).ToString(), new Font("Sans
           Serif", 10, FontStyle.Regular), linebrush, 20, x - 10);
29:  }
30:
31:  //baseline
32:  gr.DrawLine(barPen, 50, 40, 50, 545);
33:  gr.DrawLine(barPen, 50, 540, 550, 540);
```

Gambar 5. Code untuk membentuk media simulasi

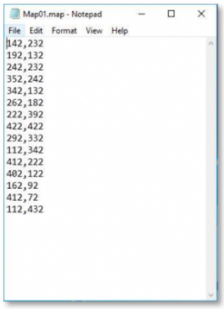
3. HASIL DAN PEMBAHASAN

Implementasi dari desain yang telah dibuat sebelumnya, menggunakan Bahasa pemrogramman C# pada Visual Studio Community 2019, menggunakan metode pengembangan perangkat lunak Extreme Programming (XP). Platform yang digunakan adalah sistem operasi Windows 10 pada komputer Intel i-7 dengan memori 16BG.

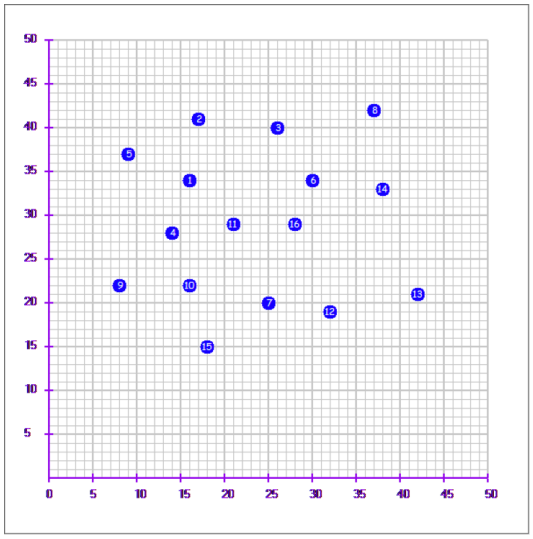
Pada gambar 6, ditampilkan hasil dari program yang dibangun dan sebagai bahan uji coba menggunakan data sederhana berupa sejumlah node dengan lokasi yang berbeda sebagai simulasi kota-kota yang akan dikunjungi, seperti diperlihatkan pada gambar 7 dan gambar 8 setelah node-node tersebut ditampilkan ke dalam program menggunakan fungsi dari tombol [Load Map].



Gambar 6. Tampilan utama program simulasi

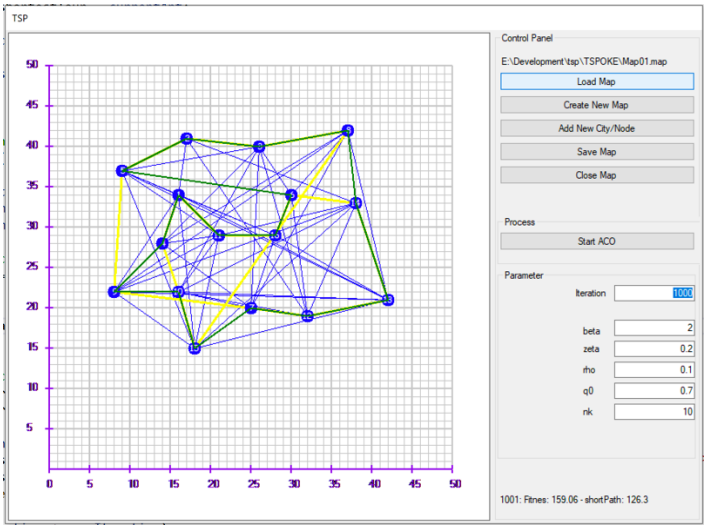


Gambar 7. Daftar posisi node atau kota sebagai uji coba masalah.



Gambar 8. Hasil simulasi posisi node/kota di dalam program simulasi

Hasil dari simulasi menggunakan metode ACO dapat dilihat pada gambar 9 berikut ini.



Gambar 9. Hasil simulasi dengan data uji coba sederhana

Pada gambar 9, dapat dilihat beberapa garis jejak (*path*) dengan warna yang berbeda, yaitu warna biru, menunjukkan garis jejak untuk setiap semut (dalam penelitian ini menggunakan 10 semut, variable *nk*), warna kuning merupakan garis jejak terbaik untuk setiap iteration (dalam penlitian ini menggunakan 1000 iterasi, variable iteration). Terakhir, warna hijau adalah garis jejak terbaik yang diperoleh dalam simulasi ini. Tabel 1 menunjukkan garis jejak terbaik yang diperoleh.

Table 1. Hasil solusi simulasi pada data uji coba sederhana.

Jenis	**Garis jejak**	**Panjang jejak**
Hijau	2-3-8-14-13-12-7-15-10-9-4-1-11-16-6-5-2	126.30
Kuning	8-10-9-7-4-11-16-14-15-6-3-1-2-5-12-13-8	159.06

Dari hasil yang didapat, dapat disimpulkan terdapat perbedaan hasil dari panjang jejak terbaik yang didapat dibandingkan dengan panjang jejak hasil terbaik pada iterasi terakhir adalah 32.76.

4. KESIMPULAN

Dari penelitian yang dilakukan dapat disimpulkan proses perancangan dan implementasi bagi model simulasi penyelesaian masalah perjalanan penjual (TSP) menggunakan metode optimisasi koloni semut (ACO) dapat diselesaikan dengan baik dan memberikan hasil yang baik.

Proses simulasi pencarian solusi jejak terpendek yang dilakukan oleh program dapat memberikan gambaran yang baik bagi pengguna dengan ditampilkannya hasil proses disetiap iterasi kepada media simulasi. Untuk setiap perubahan hasil dapat diamati dan dikonfirmasi hasilnya langsung oleh pengguna.

Untuk pengembangan, dapat juga ditampilkan matrik *pheromone* untuk menunjukkan perubahan nilai untuk setiap iterasi, bahkan akan lebih menarik kalua matrik pheromone-nya di berikan warna yang berbeda untuk setiap range nilai tertentu, sehingga pengguna dapat melihat dengan mudah, garis jejak mana yang paling menarik bagi semut, yaitu yang mempunyai nilai lebih tinggi.

Referensi

Alshayeb, Mohammad, and Wei Li. 2006. “An Empirical Study of Relationships among Extreme Programming Engineering Activities.” *Information and Software Technology* 48(11): 1068–72. http://www.sciencedirect.com/science/article/pii/S0950584906000097.

Anis, Muchlison, Siti Nandiroh, and Agustin Utami. 2007. “OPTIMASI PERENCANAAN PRODUKSI DENGAN METODE GOAL PROGRAMMING.” *Jurnal Ilmiah Teknik Industri* 5.

Candrawati, Luh Gede Ayu, and I Gusti Agung Gede Arya Kadyanan. 2017. “Optimasi Traveling Salesman Problem (TSP) Untuk Rute Paket Wisata Di Bali Dengan Algoritma Genetika.” *Jurnal Ilmiah Komputer* 10(1): 27–32.

Fojtik, Rostislav. 2011. “Extreme Programming in Development of Specific Software.” *Procedia Computer Science* 3: 1464–68. http://www.sciencedirect.com/science/article/pii/S1877050911000330.

Fortnow, Lance. 1997. ACM Sigsoft Software Engineering Notes *Nondeterministic Polynomial Time versus Nondeterministic Logarithmic Space: Time-Space Tradeoffs for Satisfiability.*

Ilwaru, Venn, Tesa Sumah, Yopi Lesnussa, and Zeth Leleury. 2017. “PERBANDINGAN

ALGORITMA HILL CLIMBING DAN ALGORITMA ANT COLONY DALAM PENENTUAN RUTE OPTIMUM." *BAREKENG: Jurnal Ilmu Matematika dan Terapan* 11: 139–50.

Khan, Indadul, and Manas Kumar Maiti. 2019. "A Swap Sequence Based Artificial Bee Colony Algorithm for Traveling Salesman Problem." *Swarm and Evolutionary Computation* 44: 428–38. http://www.sciencedirect.com/science/article/pii/S2210650216304588.

Kurniawan, T.B., Z. Ibrahim, M.F. Muhammed Saaid, and A. Yahya. 2009. "Implementation of Ant System for DNA Sequence Optimization." In *AIP Conference Proceedings*,.

Kurniawan, Tri Basuki et al. 2008. "An Ant Colony System for Dna Sequence Design Based on Thermodynamics." *Proceedings of the 4th IASTED International Conference on Advances in Computer Science and Technology, ACST 2008* (January 2008): 144–49.