Terbit online pada laman web jurnal: **https://ejurnalunsam.id/index.php/jicom/**

JICOM
Jurnal Informatika dan Teknologi Komputer

Vol. 01 No. 01 (2020) 36 - 44 ISSN Media Elektronik: 2774-7115

IMPLEMENTASI METODE HUNGARIAN UNTUK OPTIMASI BIAYA SERVIS BERBASIS WEB

Azhar[1],Yunita[2]
[1]Jurusan Teknologi Informasi dan Komputer, Politeknik Negeri Lhokseumawe
[1]azhar.tik@pnl.ac.id, yunitafajriya@gmail.com

Abstract

The matrix is the composition of a mathematical system that can assist all forms of calculations and is directly related to numbers or exact values that are large or small. The effectiveness of solving matrix problems that occur in everyday life can affect the profit side of a company, the Hungarian method is one way to solve this problem. The implementation of the Hungarian method is related to the matrix which is usually used for the case of placing an object or subject in a company. The assignment case that occurred at Bengkel Buyung Tambon-baroh relates to cost optimization for workers with their respective abilities and skills. The problem of cost optimization is very necessary considering the work schedule and the income of the workers depending on the circumstances of the institution or company concerned. Matrix calculation application using web-based Hungarian method can produce values that are considered effective for solving the problem. Based on the test results, the application produces a final value in the form of optimal costs for workers at the Buyung Tambon Baroh Workshop with their respective abilities and skills.

Keywords: matrix, square matrix, hungarian method

Abstrak

Matriks adalah komposisi dari suatu sistem matematis yang dapat membantu segala bentuk perhitungan dan berhubungan langsung dengan angka atau nilai eksak yang berjumlah besar atau kecil. Keefektifan memecahkan persoalan matriks yang terjadi dalam kehidupan sehari – hari dapat mempengaruhi sisi keuntungan dalam suatu perusahaan, Metode hungarian adalah salah satu cara memecahkan masalah tersebut. Implementasi metode hungarian berhubungan dengan matriks yang biasanya digunakan untuk kasus penempatan suatu objek maupun subjek dalam suatu perusahaan. Kasus penugasan yang terjadi di Bengkel Buyung Tambon-baroh berhubungan dengan optimalisasi biaya untuk para pekerja dengan kemampuan dan keterampilan mereka masing-masing. Masalah optimalisasi biaya sangatlah diperlukan mengingat jadwal kerja dan pedapatan para pekerja tergantung dengan keadaan lembaga atau perusahaan yang bersangkutan. Aplikasi perhitungan matriks menggunakan metode hungarian berbasis web dapat menghasilkan nilai yang dianggap efektif untuk pemecahan masalah tersebut. Berdasarkan hasil pengujian, aplikasi menghasilkan nilai akhir berupa biaya optimal untuk para pekerja pada Bengkel Buyung Tambon Baroh dengan kemampuan dan keterampilan mereka masing-masing.

Kata kunci: matriks, matriks bujur sangkar, metode hungarian

Diterima Redaksi : 14-02-2020 | Selesai Revisi : 15-03-2020 | Diterbitkan Online : 01-04-2020

1. Pendahuluan

Dalam kehidupan sehari – hari dengan mengubah persoalan ke dalam bahasa atau persamaan matematika maka suatu persoalan dapat lebih mudah diselesaikan. Tetapi terkadang suatu persoalan seringkali memuat lebih dari dua persamaan dan beberapa variabel – variabelnya. Di Negara maju sering ditemukan model ekonomi yang harus memecahkan suatu sistem persamaan dengan puluhan atau ratusan variabel.

Matriks, pada dasarnya merupakan suatu alat atau instrumen yang cukup ampuh untuk memecahkan persoalan tersebut. Matriks adalah suatu susunan bilangan yang diatur dalam baris dan kolom berbentuk persegi panjang. Penggunaan matriks dapat membentuk kumpulan analisa yang mencakup hubungan variabel – variabel dari suatu persoalan. Aplikasi matriks banyak dijumpai dalam kehidupan sehari-hari, disadari atau tidak penggunaan aplikasi tersebut banyak dimanfaatkan dalam menyelesaikan masalah-masalah yang berhubungan dengan kehidupan.

Salah satunya adalah dengan cara mengimplementasikan algoritma yang tepat, khususnya langsung berhubungan dengan matriks. Metode yang termasuk dalam metode penyelesaian masalah adalah metode hungarian .

Metode Hungarian adalah salah satu teknik pemecahan masalah untuk masalah penugasan, dimana jumlah petugas dan penugasan harus seimbang. Masalah penugasan sangat berkaitan erat dengan sejumlah data yang produktif untuk sejumlah tugas. Data yang dimiliki untuk menyelesaikan masalah penugasan adalah macam pekerjaan , jumlah karyawan dan waktu penyelesaian pekerjaan, serta penempatan biaya yang efektif untuk setiap tenaga kerja.

Keuntungan terbesar penggunaan algoritma hungarian adalah kompleksitas algoritmanya yang polinomial. Untuk suatu masalah penugasan , jumlah penugasan yang mungkin dilakukan sama dengan karena berpasangan satu-satu.

Berdasarkan pemaparan di atas tentang beberapa fungsi matriks dan metode hungarian maka penulis tertarik untuk membuat suatu aplikasi yang dapat digunakan untuk penyelesaian masalah penugasan yang akan dirangkum dalam suatu aplikasi perhitungan matriks menggunakan metode hungarian berbasis web.

Studi kasus yang akan diterapkan pada aplikasi di penelitian ini adalah pada suatu Bengkel di Tambon Baroh Aceh utara para montir atau pekerja masing – masing mempunyai tugas tersendiri menurut keterampilannya masing – masing. Biaya penugasan atau biaya pekerja mempunyai nilai yang berbeda – beda pula. Aplikasi akan membantu menguji dan memabantu mencari nilai atau nominal yang tepat serta efektif untuk setiap pekerja dalam tugasanya. Sebagaimana dijelaskan jumlah pekerja akan sama dengan jumlah penugasan, nilai – nilai tersebut akan dibuat dalam bentuk tabel / matriks dan diuji pada aplikasi. Data yang akan diuji merupakan data yang diambil langsung dari bengkel Buyung Tambon Baroh.

Aplikasi ini akan membantu pengguna untuk lebih mudah menghitung nilai dari suatu komponen berbentuk matriks bernilai besar serta memberikan informasi lebih dalam mengenai metode hungarian dalam memecahkan suatu masalah, khususnya dalam masalah penugasan atau masalah estimasi.

Penelitian ini menghasilkan nilai optimasi biaya untuk masing – masing pekerja pada kasus di Bengkel buyung Tambon Baroh dengan aplikasi perhitungan matriks menggunakan metode hungarian. Disamping itu aplikasi simulasi perhitungan matriks yang optimal dan efektif menggunakan metode hungarian agar mudah digunakan oleh semua kalangan pengguna.

Beberapa penelitian mengenai aplikasi perhitungan matriks serta metode penyelesaian masalah sebelumnya telah dibuat oleh beberapa peneliti. Di antaranya Joane Indra Prastyawan, M.J, Dewiyani Yudha Herlambang Ngunar (2008) yang membuat Aplikasi Metode Numerik dan Matriks Dalam Perhitungan Koefesien – Koefesien Regresi Linear Multiple Untuk Peramalan. Aplikasi ini memuat pembahasan mengenai operasi matriks khususnya determinan dan eliminasi Gauss Jordan yang diselesaikan dengan menggunakan aplikasi maple 10.

Putri Hayati (2013). Membuat aplikasi yang berjudul Aplikasi Perhitungan Matriks Berbasis Android. Aplikasi ini membuat perhitungan determinan matriks dengan menampilkan informasi matriks, perhitungan matriks untuk menguji kemampuan user. Aplikasi ini dibuat menggunakan bahasa pemrograman Java dan berbasis Android.

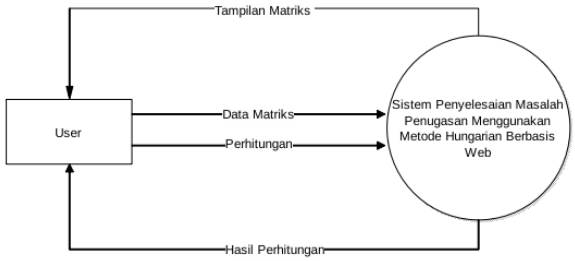
I Dewa Ketut Sanisca (2012). Melakukan penelitian dengan judul Studi Perbandingan Algoritma Genetik Dengan Metode Simpleks Yang Disempurnakan Dalam Masalah Penugasan (Assignment Problem). Penelitian yang dilakukan untuk membuktikan bahwa metode hungarian merupakan suatu metode penugasan masalah yang efektif yang bisa digunakan untuk pemecahan kasus serta membandingkannya dengan algoritma genetika.

Ahmad Zainuddin Fauzi, Entin Martiana S.Kom,M.Kom, Arna Fariza, S.Kom , M.Kom. Kom. Membuat penelitian dengan judul Pemilihan Bidang Studi Tugas Akhir Teknik Informatika Berbasis Web Menggunakan Fuzzy. Aplikasi ini dibuat menggunakan PHP dan diimplementasikan dalam bentuk web agar

dapat di akses dengan mudah oleh pengguna. Aplikiasi ini dapat membantu mahasiswa menentukan bidang studi tugas akhir berdasarkan nilai dan minat dengan hasil bidang studi yang dianjurkan.

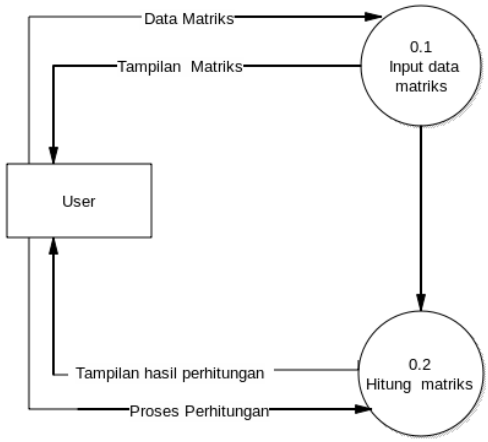
2. Metode Penelitian

DFD terdiri dari context diagram dan diagram rinci. Context diagram berfungsi memetakan model lingkungan (menggambarkan hubungan antara entitas luar, masukan dan keluaran sistem), yang direpsentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem.



Gambar 1. Perancangan Context Diagram

Pada aplikasi ini pihak yang terlibat hanya ada satu yaitu user (pengguna). User dalam sistem ini tidak ada batasan penggunaannya karena sistem ini dapat dijalankan oleh semua user. User dapat memasukkan jumlah ordo matriks dan elemen dalam matriks untuk melakukan perhitungan. Selain itu user juga dapat mengacak sendiri angka yang diinginkan dalam melakukan perhitunga matriks.



Gambar 2. Perancangan DFD Level 0

dapat dilihat bahwa DFD level 0 pada aplikasi perhitungan matriks menggunakan metode hungarian terdapat emapat buah proses yaitu :

Proses Input data matriks merupakan proses awal dalam menjalankan aplikasi yaitu user harus memasukkan jumlah ordo matriks yang diinginkan sehingga form akan menghasilkan kotak berupa baris dan kolom yang akan membantu user untuk melanjutkan proses selanjutnya, setelah user memasukkan jumlah ordo yang diinginkan , user dapat memasukkan angka atau elemen ke dalam form kolom dan baris yang telah disediakan pada aplikasi perhitungan matriks. Proses input matriks juga dapat dilakukan secara manual dimana dengan menekan tombol random pada aplikasi.

Proses perhitingan matrik adalah proses inti dari aplikasi perhitungan matriks menggunakan metode hungarian. Proses ini menentukan hasil yang akan ditampilkan oleh aplikasi serta menampilkan tahapan – tahapan penjelasan mengenai penyelesaian berupa perhitungan matriks dengan metode hungarian secara berurutan serta menampilkan hasil akhir yang akan menjadi jawaban dari persoalan awal yang dimasukkan dalam aplikasi.

Proses jalannya sistem secara keseluruhan yaitu ketika proses input angka dijalankan maka sistem menanyakan apakah user ingin memasukkan angka sendiri, jika ‘ya’ maka nilai acak angka akan keluar tetapi jika ‘tidak’ maka user akan memasukkan angka yang diinginkan sehingga sistem akan melakukan proses hitung ketika user menekan tombol submit. Proses ini jika difungsikan dalam listing program yaitu:

Program 1

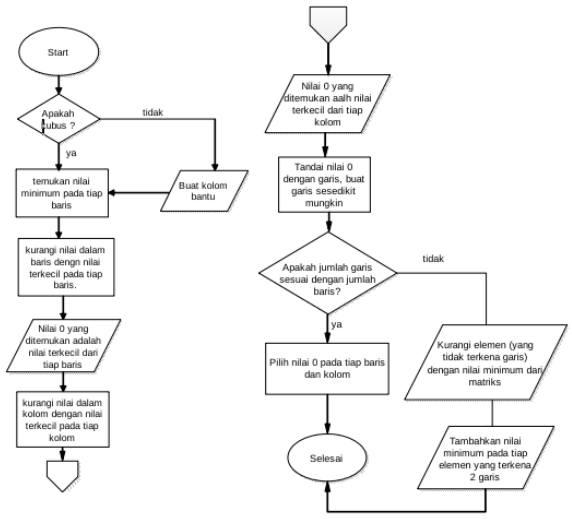
```
function random(){
for(i = 0; i < rows; i++){
      for(j = 0; j < cols; j++){
          document.getElementById("cell_r"+i+"_c"+j).value = Math.floor(Math.random()*101);
      }}}
```

Sedangkan untuk membuat fungsi tombol submit (hitung) pada listing program dapat dilihat pada kolom di bawah ini:

Program 2

```
function submit() {
    document.getElementById("random").style.display = 'block';
    cols = parseInt(document.getElementById("inputX").value);
    rows = parseInt(document.getElementById("inputY").value);
    if (neverEntered) {
        redrawTable();
    } else{ document.getElementById("bottom").style.display ='block';
    document.getElementById("theForm").style.display = 'none';
    document.getElementById("submit").style.display = 'none';
    document.getElementById("random").style.display = 'none';
    createDummyRows();}}
```

Flowchart Proses Tahapan Perhitungan



Gambar 3. Flowchart Proses Tahapan Perhitungan

3. Hasil dan Pembahasan

Hasil pengujian sistem didapatkan yaitu aplikasi sudah berjalan dengan baik, proses perhitungan matriks menggunakan matriks dengan metode hungarian dapat menghasilkan output yang sesuai dengan yang inputan yang dimasukkan serta dengan tahapan – tahapan yang sesuai dengan kaidah yang terkandung dalam metode hungarian. Di bawah ini merupakan desain tampilan serta analisa dan penjelasan mengenai proses tahapan yang dihasilkan dari aplikasi perhitungan matriks menggunakan metode hungarian. Data yang diuji untuk kasus ini merupakan data hasil berdasarkan survei ke objek yang dituju.

3.1. Pengujian aplikasi perhitungan matriks dan tampilan hasil

Perintah untuk melakukan ini sehingga kolom dan baris bisa muncul dan tampil sesuai dengan jumlah elemen (nilai) yang di masukkan adalah:

Program 3

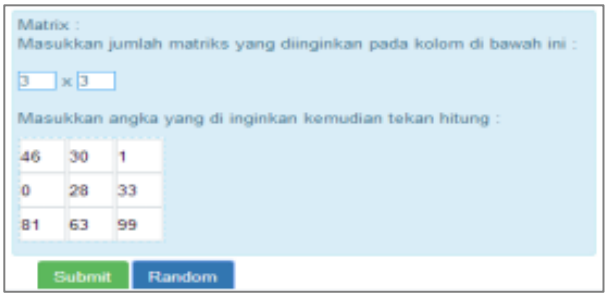
```
function readFirstInput(num, xory)
{   if (parseInt(num) == num)
    {
        if (xory == 'x' && num < 125) cols = num;
        if (xory == 'y' && num < 125) rows = num;
        redrawTable();
    }}
```

Fungsi di atas adalah membaca input ketika angka di masukkan:

Program 4

```
function redrawTable(){
    neverEntered = false;
    innerString = "<table>";
    for (var i = 0; i < rows; i++) {
        innerString+="<tr>";
        for (var j = 0; j < cols; j++) {
        innerString+="<td><input type = \"text\" class = \"generatedInput\" id = \"cell_r"+i+"_c"+j+"\" /></td>";}
        innerString+="</tr>";}
    innerString+="</table>";
    document.getElementById("mainTable").innerHTML=innerString;
    document.getElementById("text2").innerHTML="Masukkan angka yang di inginkan kemudian tekan hitung :";}
```

Fungsi ini bertujuan untuk membuat tabel baru yang muncul sesuai dengan nilai yang dimasukkan pada tabel di atas sebelumnya. Langkah selanjutnya memasukkan elemen pada kolom matriks yang telah disediakan.



Gambar 4. Tampilan Halaman Input Random Matriks

Pada gambar diatas angka dimasukkan dengan menekan tombol random, elemen yang dimasukkan merupakan elemen acak yang teratur otomatis di dalam aplikasi.

Fungsi tombol tahapan proses pada listing program adalah:

Program 5

```
function stepForward()
{ // menuju satu langkah berikutnya
    if (userStep+1 <= maxStep) {
        showStep(userStep+1);
        userStep += 1;  }
}
function stepBack()
{ // menuju satu langkah sebelumnya
    if(userStep-1 >= 1) {
        showStep(userStep-1);
        userStep -= 1;
function fastForward()
{ // langsung menuju langkah terakhir
    showStep(maxStep);
    userStep = maxStep;}
function rewind()
{ // langsung menuju langkah pertama
    showStep(1); userStep = 1;}
```

Perhitungan matriks dengan metode hungarian dapat dilihat pada gambar serta penjelasan di bawah ini:

Gambar 5. proses mencari elemen terendah pada baris

Tampilan diatas menjelaskan bahwa proses sudah berjalan pada tahap selanjutnya, yaitu mencari elemen minimum pada tiap baris. Fungsinya pada listing program dapat dilihat pada kolom di bawah ini:

Program 6

```
var borderMod = new Array();
    for (i = 0; i < rows; i++) {
        borderMod[i] = new Array();
        for (j = 0; j < cols; j++) {
            borderMod[i][j] = 0;
        }}
    var lowestHolder = new Array();
    currentLowest = largest;
    for (i = 0; i < rows; i++) {
            // temukan nilai terendah pada tiap baris
        for (j = 0; j < cols; j++) {
            if
(parseInt(document.getElementById("cell_r"+i+"_c"+j).value) <=
currentLowest) {
                currentLowest = parseInt (document.getElementById
("cell_r"+i+"_c"+j).value);
                currentLowestIndex = j;
            }
        }
        borderMod[i][currentLowestIndex] = 1;
        lowestHolder[i] = currentLowest;
        currentLowest = largest;
    }
    text = "Temukan <div class = \"colorMod_1\"> nilai minimum
</div> pada tiap baris.";
    steps[stepCreation] = new step(true, stepCreation, 1, text, holder,
borderMod);
```

Proses selanjutnya adalah mengurangkan elemen minimum pada tiap baris untuk itu digunakan fungsi di bawah ini:

Program 7

```
var holder_2 = new Array();
    for (i = 0; i < rows; i++) {
        holder_2[i] = new Array();
        for (j = 0; j < cols; j++) {
            holder_2[i][j] = 0;
        }
    }
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            holder_2[i][j] = steps[stepCreation-1].grid[i][j] -
lowestHolder[i];
        }
    }
    text = "Kurangi nilai dalam baris dengan <div class =
\"colorMod_1\">nilai minimum</div>. Nilai <div class =
\"colorMod_1\">nol yang ada</div> adalah yang terendah dari tiap
baris.";
    steps[stepCreation] = new step(true, stepCreation, 1, text,
holder_2, borderMod);
```

Gambar 6. Proses pengurangan elemen pada baris

Tampilan diatas menunjukkan proses aplikasi untuk langkah 3 pada proses perhitungan.

Langkah selanjutnya adalah mencari elemen terendah pada tiap kolom dan mengurangkannya seperti proses yang dilakukan pada langkah 2 dan langkah 3. Selanjutnya tampilan akan terlihat seperti gambar dibawah:

Gambar 7. Mencari elemen minimum pada tiap kolom

Fungsi untuk mencari elemen terendah pada tiap kolom dan mengurangkannya :

Program 8

```
var borderMod = new Array();
    for (i = 0; i < rows; i++) {
        borderMod[i] = new Array();
        for (j = 0; j < cols; j++) {
            borderMod[i][j] = 0;
        }
    }
    var lowestHolder = new Array();
    currentLowest = largest;
    for (j = 0; j < cols; j++) {
        for (i = 0; i < rows; i++) {
            if (steps[stepCreation-1].grid[i][j] <= currentLowest) {
                currentLowest = steps[stepCreation-1].grid[i][j];
                currentLowestIndex = i;
            }
        }
        borderMod[currentLowestIndex][j] = 1;
        lowestHolder[j] = currentLowest;
        currentLowest = largest;
    }
    text = "Temukan <div class = \"colorMod_2\">nilai
minimum</div> pada tiap kolom.";
    steps[stepCreation] = new step(true, stepCreation, 2, text,
holder_2, borderMod);
    var holder_3 = new Array();
    for (i = 0; i < rows; i++) {
        holder_3[i] = new Array();
        for (j = 0; j < cols; j++) {
            holder_3[i][j] = 0;
        }
    }
    for (j = 0; j < cols; j++) {
        for (i = 0; i < rows; i++) {
            holder_3[i][j] = steps[stepCreation-1].grid[i][j] -
lowestHolder[j];
        }
    }
    text = "Kurangi nilai kolom dengan <div class =
\"colorMod_2\">nilai minimum</div>. Nilai <div class =
\"colorMod_2\">nol yang keluar</div> adalah nilai terendah dari
tiap kolom.";
    steps[stepCreation] = new step(true, stepCreation, 2, text,
holder_3, borderMod);
    var holder_6 = new Array();
    for (i = 0; i < rows; i++) {
        holder_6[i] = new Array();
        for (j = 0; j < cols; j++) {
            holder_6[i][j] = 0;
        }
    }
```

Selanjutnya jika elemen nol sudah muncul di tiap baris dan kolom, pada masing – masing kolom dan baris tandai dengan garis seperti pada gambar dibawah:

Gambar 8. Proses menandai elemen 0 dengan garis

3.2. Analisa dan pengujian kasus

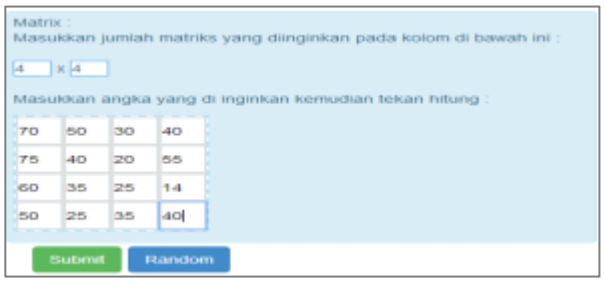
Setelah melakukan pengujian proses perhitungan dalam aplikasi dan program, kini implementasi kasus dalam aplikasi harus dilakukan agar dapat membuktikan bahwa aplikasi sudah sesuai dengan perancangan. Kasus yang di terapkan dalam aplikasi adalah salah satu kasus yang terjadi pada suatu bengkel di Tambon Baroh. Permasalahan yang diangkat adalah kasus optimalisasi biaya atau penyesuaian gaji pekerja dengan keterampilan mereka masing – masing. Optimalisasi biaya untuk masing – masing pekerja (montir) montir yang sesuai dengan keterampilan mereka masih belum belum efektif. Diharapkan dengan adanya pengujian kasus pada aplikasi ini pemberian gaji untuk masing – masing pekerja dapat sesuai dengan semestinya. Aplikasi akan menguji data dan menghasilkan nilai sekaligus membantu penyesuaian biaya montir (pekerja).

Pada kasus ini 4 montir diambil sebagai objek. Empat pekerjaan yang berbeda diselesaikan oleh 4 (empat) pekerja. Sesusai ketentuan yang ditetapkan jumlah pekerja harus sama dengan jumlah pekerjaannya. Masing – masing pekerja mendapatkan tugas serta jadwal yang berbeda – beda pula diantaranya mengecat dan las mobil, tugas perbaikan mesin, membeli peralatan – peralatan yang dibutuhkan dan lain – lain. Biaya penugasan seorang montir juga berbeda-beda. Sesuai dengan tingkat keterampilan masing-masing pekerja. Biaya penugasan (angka dalam tabel dinominalkan dalam puluhan ribu rupiah) untuk macam-macam pekerjaan ditunjukkan pada Tabel 4.1. berikut :

Tabel 3. Tabel data biaya kasus optimalisasi

MONTIR	TUGAS 1	TUGAS 2	TUGAS 3	TUGAS 4
A	70	50	30	40
B	75	40	20	55
C	60	35	25	14
D	50	25	35	50

Matriks yang akan dibentuk adalah berorodo 4, artinya matriks berjumlah 4 baris dan 4 kolom. Nilai yang tertera dalam tabel akan dimasukkan ke dalam aplikasi seperti gambar di bawah ini :



Gambar 8. Tampilan Input Nilai Tabel

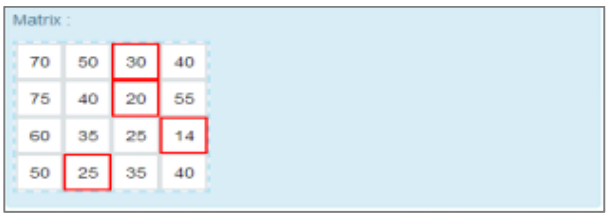
Kemudian akan diuji dengan proses perhitungan dengan tombol submit (hitung). Langkah-langkah penyelesaian dalam kasus ini adalah sebagai berikut :

Merubah matriks biaya menjadi Opportunity Cost. Ini dicapai dengan memilih elemen terkecil dari setiap baris dari matriks biaya mula-mula untuk mengurangi seluruh elemen setiap baris. Dari Tabel Reduced Cost matriks sebagai berikut:

Tabel 4. Langkah pertama penyelesaian

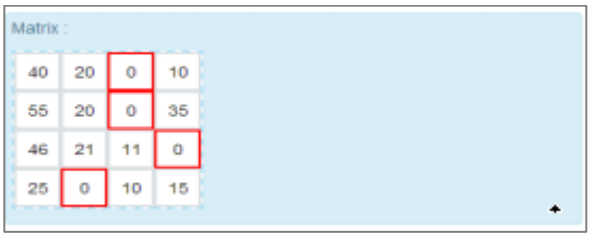
MONTIR	TUGAS 1	TUGAS 2	TUGAS 3	TUGAS 4
A	40	20	0	10
B	55	20	0	35
C	46	21	11	0
D	25	0	10	15

Pada aplikasi tampilan program untuk proses pertama terlihat seperti pada gambar 9:



Gambar 9. Proses cari elemen terkecil

Kemudian elemen terkecil akan dikurangkan pada tiap baris dan akan muncul tampilan seperti ini gambar 10:



Gambar 10. Proses pengurangan elemen pada baris

Hasilnya sama seperti pada tabel 4. Jika dituliskan dengan fungsi matematis akan menghasilkan nilai yang sama:

Elemen terkecil pada matriks bari pertama di atas adalah 30, maka tiap nilai dalam baris dikurangkan dengan 15:

(70 – 30 = 40) (50 -30 = 20) (30 – 30 = 0) (40– 30 = 10)

Sedangkan pada baris kedua adalah 20, maka tiap elemen dalam baris dikurangkan dengan 15 :

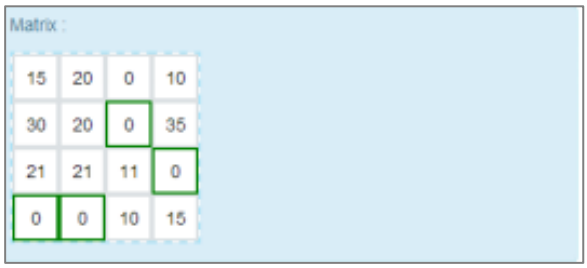
(75– 20 = 55) (40 – 20 = 20) (20 – 20 = 0) (55 - 20 = 35)

Hasil yang dihasilkan adalah sama begitupun dengan proses perhuitungan selanjutnya. Kemudian langkah ke 2 untuk penyelesaian, Reduced Cost Matrik di atas terus dikurangi untuk mendapatkan total opportunity cost matrik. Hal ini dapat dicapai dengan memilih elemen terkecil dari setiap kolom pada Reduced Cost Matrik untuk mengurangi seluruh elemen dalam kolom-kolom tersebut. Matriks total opportunity cost ditunjukkan dalam Tabel 5:

Tabel 5. Matrik total opportunity cost

MONTIR	TUGAS 1	TUGAS 2	TUGAS 3	TUGAS 4
A	15	20	0	10
B	30	20	0	35
C	21	21	11	0
D	0	0	10	5

Pada aplikasi untuk proses di atas juga menghasilkan nilai 0 pada tempat yang sama. Dapat dilihat pada gambar 11:



Gambar 11. Proses pengurangan elemen kecil pada kolom

Langkah selanjutnya dalam pengujian kasus ini adalah mencari nilai suatu pekerjaan dengan suatu total opportunity cost nol. Untuk mencapai penugasan ini dibutuhkan 4 "independent/cros" (karena ada 4 pekerja atau karyawan) dalam matrik. Ini berarti setiap karyawan harus ditugaskan hanya untuk suatu pekerjaan dengan opportunity cost sama dengan nol. Atau setiap pekerjaan harus diselesaikan hanya oleh satu karyawan. Prosedur praktis untuk melakukan tes optimalisasi adalah dengan menarik sejumlah minimum garis horizontal atau vertikal untuk meliputi seluruh elemen bernilai nol dalam total opportunity cost matrik. Bila jumlah garis sama dengan baris dan kolom, penugasan optimal adalah layak. Bila tidak sama maka matrik harus direvisi.

Tabel 6. Tabel tes optimalisasi

MONTIR	TUGAS 1	TUGAS 2	TUGAS 3	TUGAS 4
A	15	20	0	10
B	30	20	0	35
C	21	21	11	0
D	0	0	10	5

Dalam Tabel 6 ada tiga baris yang meliputi seluruh nilai nol dibanding empat baris atau kolom, sehingga langkah berikutnya untuk merevisi matriks.

Ini juga tejadi pada aplikasi ketika proses selanjutnya dijalankan seperti ditunjukkan pada gambar 12:

Gambar 12. Penarikan garis pada elemen 0

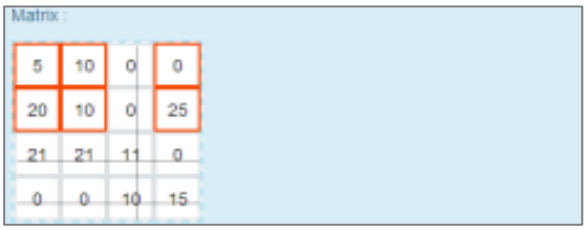
Karena jumlah garis belum sesuai dengan jumlah baris, maka untuk merevisi total opportunity cost matrik, pilih elemen terkecil yang belum terliput garis-garis (yaitu opportunity cost terendah, atau pada contoh diatas = 10) untuk mengurangi seluruh elemen yang belum terliput. Kemudian tambahkan dengan jumlah yang sama (nilai elemen terkecil) pada seluruh elemen-elemen yang mempunyai dua garis yang saling bersilangan (11 pada baris C dan 10 pada baris D). Jadi hasil perbaikannya adalah 21 pada baris C dan 20 pada baris D. Matriks yang telah direvisi dapat dilihat pada tabel 7 yang didapat dengan mengikuti prosedur diatas.

Tabel 7. Tabel matriks baru dan optimalisasi nilai

MONTIR	TUGAS 1	TUGAS 2	TUGAS 3	TUGAS 4
A	5	10	0	0
B	20	10	0	25
C	21	21	21	4
D	0	0	20	15

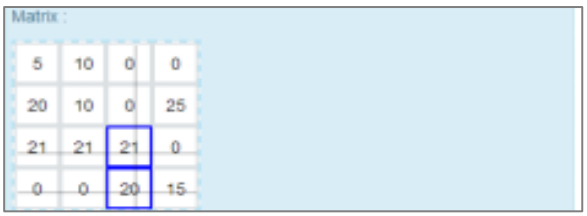
Jika pada pengujian kasus merevisi matriks dijalankan dengan cara di atas, pada aplikasi juga diselesaikan dengan cara yang sama. Gambar di bawah ini menjelasakan jalannya proses:

Pertama jika garis yang ditarik belum sesuai dengan jumlah baris maka kurangi elemen yang belum terkena garis dengan elemen terkecil dari matriks, yaitu 5. Hasilnya seperti ditunjukkan pada gambar 13:



Gambar 13. Proses pengurangan nilai yang tidak terkena garis

kemudian jumlahkan elemen yang terkena 2 garis dengan elemen terkecil dari matriks yaitu 5, hasilnya seperti yang terlihat pada gambar 14:



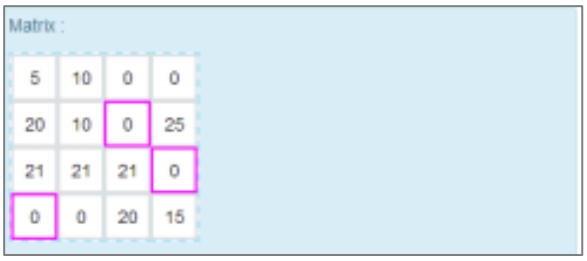
Gambar 14. Proses penjumlahan elemen yang terkena garis 2 kali

Maka akan muncul matriks baru dan penarikan garis untuk elemen 0 diulang kembali, tampilan yang dihasilkan seperti gambar 15:

Gambar 15. Proses penarikan garis kembali menandai elemen 0

Selanjutnya dalam tabel dibutuhkan minimal empat garis untuk meliputi seluruh nilai nol atau sama dengan jumlah baris atau kolom, sehingga matrik penugasan optimal telah tercapai. Tampilan aplikasi juga menghasilkan hasil yang sama yaitu kini jumlah garis sudah sesuai dengan jumlah baris.

Kini matriks penugasan telah tercapai dan elemen yang diambil adalah satu elemen 0 dari tiap – tiap baris dan kolom yaitu seperti yang ditunjukkan pada gambar 26.



Gambar 16. Penemuan nilai efektif sesuai dengan tabel pengujian

Biaya penugasan (angka dalam tabel dinominalkan dalam puluhan ribu rupiah) optimal biaya keterampilan masing – masing montir adalah sebagai berikut :

Tabel 8. Tabel hasil optimalisasi biaya pekerja

JADWAL PENUGASAN	BIAYA
A-TUGAS 3	Rp 30
B-TUGAS 3	Rp 20
C-TUGAS 4	Rp 14
D-TUGAS 1	Rp 50

Maka hasil akhir yang ditemukan pada aplikasi untuk kasus ini adalah nilai atau nominal untuk masing – masing karyawan (pekerja). Nilai yang ditunjukkan oleh tabel 8 bahwa untuk pekerja A yang sesuai dengan tugas dan keterampilannya mendapatkan nilai efektif sebesar 30 untuk satu kali pekerjaan, untuk pekerja B dapat dilihat nilai yang didapatkan adalah 20 untuk tugas dan keterampilannya, pekerja C mendapat biaya yang optimal sebesar 14 sesuai dengan keterampilannya begitupun dengan pekerja D yang mendapatkan nilai terbesar pada hasil pengujian kasus optimal biaya montir di Bengkel Buyung Tambon Baroh. Elemen nol yang dihasilkan oleh aplikasi merupakan nilai efektif dari pemecahan kasus. Nilai efektif awal berpedoman pada tabel 3 yang merupakan data yang diuji.

4. Kesimpulan

Berdasarkan hasil implementasi dan analisa terhadap aplikasi maka dapat disimpulkan bahwa :

1. Pengujian kasus optimasi biaya montir di Bengkel Buyung Tambon Baroh mendapatkan hasil akhir berupa nilai efektif dari data yang diuji.

2. Masing- masing pekerja mendapatkan nilai biaya yang sesuai dengan data awal yang diuji yaitu pekerja A mendapatkan nilai 30, pekerja B mendapatkan nilai 20, pekerja C mendapatkan nilai 14 dan pekerja 4 mendapatkan nilai 50 (nilai yang didapat oleh masing – masing pekerja dinominalkan dalam puluhan ribu rupiah).

3. Aplikasi dapat menguji dan menyelesaikan masalah penugasan pada kasus optimasi biaya pekerja Bengkel Buyung Tambon Baroh dengan baik pada Web server.

4. Data yang diuji pada aplikasi harus sesuai, artinya jumlah pekerja sama dengan jumlah pekerjaan yang ditugaskan.

Daftar Rujukan

[1] Susanto, Alvin. Penggunaan Algoritma Hungarian dalam menyelesaikan persoalan matriks berbobot. 2006. Program studi Teknik Informarika, ITB (Institut Teknologi Bandung). Bandung.

[2] Budi, Sutedjo Dharma Oetomo. 2002. Perencanaan dan Pembangunan Sistem Informasi. Erlangga, Jakarta.

[3] Entireprise, Jubilee. 2008. 68 Trik Rahasia Joomla. Penerbit PT Elex Media Komputindo, Jakarta.

[4] G.Ayorkor, Mills Tattey, Anthony Stent, M.Barradine. 2007. The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs.

[5] Henry, Bustami. 2005.Prinsip – Prinsip Riset Operasi. Erlangga, Jakarta.

[6] J.Booth. K.A Stroud. 2003. Matematika Teknik. Penerbit Erlangga, Jakarta.

[7] Steven J. Leon. 2001. Aljabar Linear dan Aplikasinya. Erlangga, Jakarta.

[8] Meissa, Indra.2009. Bikin Website Asik Ala Joomla 1.5. Gagas Media, Bandung.

[9] Simarmata, Janner. 2010. Rekayasa WEB. ANDI OFFSET.Yogyakarta.

[10] Sibero, Alexander F.K. 2012. "Kitab Suci WEB Programming". Mediakom, Jakarta.

[11] Zaenal, Ali. 2011 Cepat dan Mudah Membuat Website Keren dengan Wordpress 3.x.. Media kita, Jakarta.

[12] Joane Indra Prastyawan, M.J, Dewiyani Yudha Herlambang Ngunar.2008. Aplikasi Metode Numerik dan Matriks Dalam Perhitungan Koefesien – Koefesien Regresi Linear Multiple Untuk Peramalan.

[13] Hayati, Putri. 2013. Aplikasi Perhitungan Matriks Berbasis Android.

[14] I Dewa Ketut Sanisca.2012. Studi Perbandingan Algoritma Genetik Dengan Metode Simpleks Yang Disempurnakan Dalam Masalah Penugasan (Assignment Problem).

[15] Ahmad Zainuddin Fauzi, Entin Martiana S.Kom,M.Kom, Arna Fariza, S.Kom , M.Kom. Kom. Membuat penelitian dengan judul Pemilihan Bidang Studi Tugas Akhir Teknik Informatika Berbasis Web Menggunakan Fuzzy