APLIKASI METODE HUNGARIAN DALAM PENUGASAN PEKERJA

Faatulo Gate[1], Jeremia Siregar[2] Rikardo H Siahaan[3]

Mahasiswa Teknik Informatika, Fakultas Teknologi Industri[1] Dosen Teknik Informatika,
Fakultas Teknologi Industri[2] Institut Sains Dan Teknologi TD Pardede
Jl. DR. TD. Pardede No.8 Medan 20153

Email : tulniagatsyadoh@gmail.com[1], jeremiasiregar@istp.ac.id[2] rikardosiahaan@istp.ac.id[3]

ABSTRAK

Model penugasan dapat diilustrasikan sebuah proyek yang dapat dibagi menjadi sejumlah pekerjaan yang harus diselesaikan dan terdapat sekelompok pekerja yang mampu untuk menyelesaikan setiap jenis pekerjaan yang ada, tetapi dengan biaya yang berbeda-beda, maka hal yang harus dilakukan adalah menentukan dengan tepat kepada siapa pekerjaan tersebut dapat diserahkan sedemikian sehingga jumlah biaya yang dikeluarkan sekecil mungkin. Salah satu metode yang dapat digunakan adalah metode Hungarian. Jumlah n entri dari sebuah penugasan disebut biaya (cost) penugasan tersebut. Penugasan dengan biaya terkecil yang mungkin disebut penugasan optimal (optimal assigment). Metode penugasan dengan menggunakan metode Hungarian akan menetapkan terlebih dahulu matriks biaya atau dikenal juga dengan cost matrix. Dari matriks yang dihasilkan akan dievaluasi apakah sudah optimal atau belum. Jika ternyata belum optimal, maka akan direvisi atau perbaikan tehadap matriks tersebut. Demikian matriks baru ini akan dievaluasi kembali sampai diperoleh bentuk optimalnya. Dari matriks yang sudah optimal ini maka dapat dilakukan penentuan pekerja yang tepat untuk sejumlah pekerjaan dengan total biaya seminimal mungkin.

Kata Kunci : Metode Hungarian, Matriks, Visual Basic

ABSTRACT

The assignment model can be illustrated as a project that can be divided into a number of jobs that must be completed and there is a group of workers who are able to complete each type of work, but at different costs, so the thing that must be done is to determine exactly who the work can be done to. submitted in such a way that the total costs incurred are as small as possible. One method that can be used is the Hungarian method. The number of n entries from an assignment includes the cost of the assignment. The assignment with the smallest possible cost is called optimal assignment. The assignment method using the Hungarian method will first set the cost matrix or also known as the cost matrix. The resulting matrix will be evaluated whether it is optimal or not. If it turns out that it is not optimal, then revisions or improvements will be made to the matrix. For example, this new matrix will be re-evaluated until its optimal form is obtained. From this optimal matrix, it is possible to determine the right workers for a number of jobs with the minimum possible total cost.

Keywords: *Hungarian Method, Matrix, Visual Basic*

1. PENDAHULUAN

Model penugasan merupakan kasus khusus dari model transportasi dalam bidang ilmu Teknik Riset dan Operasi dimana sejumlah "m" sumber ditugaskan kepada sejumlah "n" tujuan (satu sumber untuk satu tujuan) sedemikian sehingga didapat ongkos total yang minimum. Dalam hal ini, sumber identik dengan pekerjaan dan tujuan identik dengan mesin-mesin (pekerja). Model penugasan dapat di ilustrasikan sebagai berikut, apabila sebuah proyek dapat dibagi menjadi sejumlah pekerjaan yang harus diselesaikan dan terdapat sekelompok pekerja yang mampu untuk menyelesaikan setiap jenis pekerjaan yang ada, tetapi dengan biaya yang berbeda- beda, maka hal yang harus dilakukan adalah menentukan dengan tepat kepada siapa pekerjaan tersebut dapat diserahkan sedemikian sehingga jumlah biaya yang dikeluarkan sekecil mungkin. Salah satu metode yang dapat digunakan adalah metode *Hungarian*.

Jumlah n entri dari sebuah penugasan disebut biaya (*cost*) penugasan tersebut. Penugasan dengan biaya terkecil yang mungkin disebut penugasan optimal (*optimal assignment*). Metode penugasan dengan menggunakan metode *Hungarian* akan menetapkan terlebih dahulu matriks biaya atau dikenal juga dengan *cost matrix*. Dari matriks yang dihasilkan akan dievaluasi apakah sudah optimal atau belum. Jika ternyata belum optimal, maka akan dilakukan revisi atau perbaikan terhadap matriks tersebut. Demikian matriks baru ini akan dievaluasi kembali sampai diperoleh bentuk optimalnya. Dari matriks yang sudah optimal ini maka dapat dilakukan penentuan pekerja yang tepat untuk sejumlah pekerjaan dengan total biaya seminimal mungkin.

Berdasarkan latar belakang di atas, maka penulis merancang sebuah perangkat lunak yang dapat melakukan penugasan pekerjaan yang optimal dengan menggunakan metode *Hungarian*. Pengerjaan perangkat lunak ini akan diberi judul 'Aplikasi Metode *Hungarian* Dalam Penugasan Pekerja'.

2. TINJAUAN PUSTAKA

2.1. PERANGKAT LUNAK

Perangkat lunak (*software*) dapat diartikan sebagai program komputer, struktur data, dan dokumentasi yang berkaitan, yang menyediakan metode logika, prosedur atau kontrol yang diminta. Perangkat lunak (*software*) memiliki beberapa karakteristik yang membedakannya dengan perangkat keras (*hardware*) antara lain :

a. Perangkat lunak dikembangkan dan direkayasa, bukan dirakit seperti perangkat keras. Meskipun ada beberapa kesamaan pengertian antara kedua istilah tersebut, tetapi pada dasarnya berbeda.
b. Perangkat lunak dibuat berdasarkan sistem yang sifatnya berbeda – beda, sedangkan perangkat keras dibuat berdasarkan rakitan komponen yang sudah ada.
c. Perangkat lunak tidak bisa rusak kecuali adanya kesalahan dari pembuat program (*programmer*), sedangkan tingkat kerusakan perangkat keras sangat tinggi.

2.2. PERANGKAT LUNAK PEMBELAJARAN

Seiring dengan perkembangan peradaban manusia dan kemajuan pesat di bidang teknologi, tanpa disadari komputer telah ikut berperan dalam dunia pendidikan terutama penggunaannya sebagai alat bantu pengajaran. Percobaan penggunaan komputer untuk proses belajar dimulai di Amerika Serikat pada akhir tahun 1950-an dan awal tahun 1960-an. Kemudian penelitian selanjutnya dilakukan oleh Harvard University bekerja sama dengan IBM pada tahun 1965. Setelah munculnya komputer mikro, sistem pengajaran dengan komputer menjadi semakin meluas pada pengembangan aplikasi perangkat lunak ajar yang dikenal dengan istilah perangkat lunak pembelajaran. Perangkat lunak pembelajaran dengan komputer muncul dari sejumlah disiplin ilmu, terutama ilmu komputer dan psikologi. Dari ilmu komputer dan matematika muncul program – program yang membuat semua perhitungan dan

fungsi lebih mudah dan bermanfaat. Sedangkan dari ilmu psikologi muncul pengetahuan mengenai teori belajar, teknik belajar, serta motivasi yang baik.

2.3. MULTIMEDIA

Menurut Burger (1993) Multimedia adalah penggabungan dari dua atau lebih media yang berbeda dengan komputer, dengan kata lain multimedia adalah kemampuan yang memungkinkan penggabungan teks, grafik, video, animasi dan suara berbasiskan komputer.

Multimedia membuat proses membaca menjadi dinamis dengan memberikan suatu dimensi baru yang penting terhadap kata-kata. Di dalam menyampaikan makna, kata-kata di dalam multimedia bertindak sebagai pemicu yang memungkinkan penggunanya memperluas teks untuk mempelajari suatu topik dengan lebih dalam dan menjadikan lebih hidup dengan suara, gambar, musik, dan video. Penggunaan unsur multimedia dimaksudkan untuk membuat suatu perangkat lunak lebih menarik sehingga seseorang akan lebih antusias untuk menggunakan perangkat lunak tersebut.

2.4. MATRIKS

Matriks adalah suatu susunan bilangan-bilangan yang berbentuk empat persegi panjang. Bentuk umum matriks yang mempunyai *m* baris dan *n* kolom adalah

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & & a_{mn} \end{pmatrix}$$

dan disebut matriks *m* x *n (*dibaca "*m* kali *n*").

Huruf-huruf besar A, B, … menunjukkan suatu matriks, sedang huruf-huruf kecil a, b, … menunjukkan suatu bilangan, dan disebut *scalar.* Dua matriks A dan B dikatakan sama, ditulis A = B, jika keduanya mempunyai ukuran sama, yaitu banyaknya baris dan kolom sama, dan memenuhi unsur-unsur yang letaknya sama. Oleh karena itu kesamaan dua matriks *m* x *n* ekivalen dengan sistem persamaan yang terdiri atas *mn* buah persamaan, dengan masing-masing unsur yang letaknya sama memberikan suatu persamaan.

2.5. METODE HUNGARIAN

Harold W. Kuhn dalam karya ilmiahnya yang berjudul "*The Hungariann Method for the assignment problem*" mendeskripsikan sebuah algoritma untuk mengkonstruksikan sebuah graf berbobot maksimum. Dalam karya ilmiahnya tersebut, Kuhn menjelaskan bagaimana cara kerja algoritma yang ditemukan oleh dua orang ahli matematika berkebangsaan Hungarian yang bernama D. Konig dan E. Egervary pada tahun 1931 tersebut, telah berkontribusi terhadap algoritma yang ditemukannya tersebut. Hal ini juga yang dijadikan alasan baginya mengapa algoritma yang ditemukannya tersebut diberi nama metode Hungariann.

Egervary menggunakan teorema Konig sebagai sebuah sub prosedur untuk membuktikan teknik yang digunakannya. Sedangkan, ide utama dari algoritma Kuhn adalah dua bagian terpisah yang terdapat dalam pembuktian Egervary dikombinasikan menjadi satu. Jasa utama dari metode Hungarian yang ditemukan oleh Kuhn adalah bahwa selama setengah abad lamanya, algoritmanya menjadi titik awal dari perkembangan yang sangat pesat dari algoritma kombinasi yang efisien, yang sekarang dikenal dengan nama Optimisasi Kombinatorial (*Combinatorial Optimization*) Ide Kuhn yang dikembangkan dari *assignment problem* ini telah diaplikasikan oleh L.R. Ford dan D.R. Fulkerson dalam masalah transportasi dan secara umum untuk meminimalkan arus biaya. Salah satu algoritma lainnya yang menerapkan ide dari Kuhn adalah *Edmonds' weighted matroid intersection algorithm.*

2.6. STRUKTUR DATA ARRAY (LARIK)

Struktur berarti susunan/jenjang, dan data berarti sesuatu simbol/huruf/lambang angka yang menyatakan sesuatu. Struktur data berarti susunan dari simbol/ huruf/lambang angka untuk menyatakan sesuatu hal. Gabungan dari algoritma dan struktur data akan membentuk suatu

program.

Data terstruktur adalah suatu tipe data yang batas nilainya masih dapat diuraikan, contohnya antrian, set, tumpukan (stack), dan sebagainya. Data terstruktur terbagi dua yaitu:

a. Data Terstruktur Linier (Struktur Data Linier).
   Contoh struktur data linier adalah larik (*array*), tumpukan (*stack*), rekaman (*record*), antrian, *linked list* dan sebagainya.
b. Data Terstruktur Non Linier (Struktur Data Non Linier)
   Contohnya adalah *graph*, *tree* (pohon), dan sebagainya.

Array (larik) adalah suatu tipe data terstruktur yang dapat menampung (berisikan) suatu data yang sejenis. Komponen – komponen dari *Array*, yaitu nama *Array*, nilai *Array*, index *Array*, dan jenis (tipe) *Array*.

Array terdiri dari beberapa jenis antara lain:

- Array 1 dimensi
  Contohnya,
  A[1] = 'A', A[2] = 'B', A[3] = 'C', A[4] = 'D'
- *Array* Multi dimensi (≥ 2)
  Contoh, misalkan diketahui sebuah *Array* dua dimensi B dengan jenis data *Integer*,
  B[1,1] = 1 B[1,2] = 2
  B[2,1] = 3

3. METODE PENELITIAN

3.1. Metode Pengembangan Sistem

Metode yang digunakan dalam pengembangan aplikasi penilaian kinerja yaitu metode *waterfall.* Metode *waterfall* adalah model klasik yang bersifat sitematis, berurutan dalam membangun software. Model ini dilakukan pendekatan secara sistematis dan berurutan. Disebut *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Menurut (Bassil, 2017) model waterfall SDLC (System Development Life Cycle) adalah sebuah metodologi untuk merancang dan membangun sistem perangkat lunak, yaitu proses perancangannya bertahap mengalir semakin ke bawah (mirip dengan air terjun).

3.2. Perancangan

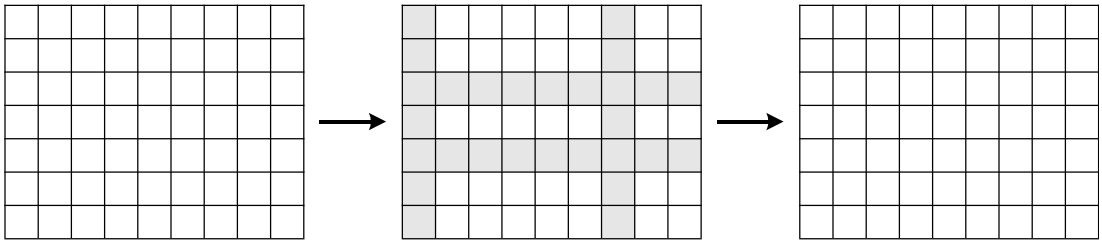
a. Penggunaan Tools

Perangkat lunak pembelajaran penugasan pekerja dengan metode *Hungarian* ini dirancang dengan menggunakan bahasa pemrograman *Microsoft Visual Basic* 6.0 dan didukung oleh beberapa objek dasar berikut:

1. *Label*, yang digunakan untuk menampilkan keterangan.
2. *Shape*, yang digunakan untuk dekorasi tambahan.
3. *MSFlexgrid*, yang digunakan untuk menginput data matriks biaya dan menampilkan matriks hasil perhitungan.
4. *Combobox*, yang digunakan untuk melakukan pemilihan ordo matriks dan solusi.
5. *Textbox*, yang digunakan untuk menampilkan data total solusi.
6. *Common dialog control*, yang digunakan untuk menampilkan dialog *save* dan *open*.
7. *Command button*, digunakan sebagai tombol eksekusi.
8. *PictureBox*, yang digunakan untuk menampilkan gambar.
9. *Image*, yang digunakan untuk menampilkan gambar panah.

b. Perancangan Animasi

Perangkat lunak ini menggunakan teknik animasi pergantian warna. Teknik ini digunakan pada proses perhitungan metode Hungarian yaitu pada saat penarikan garis-garis yang melalui baris-baris dan kolom-kolom yang sesuai sehingga seluruh entri-entri nol matriks biaya ini dapat tertutup dan jumlah garis-garis yang digunakan adalah minimum. Baris-baris dan kolom-kolom yang dipilih pada tabel akan diganti warna latarnya. Proses yang dilakukan dapat dilihat pada ilustrasi gambar 1 berikut:



Keadaan awal:
Warna latar dari setiap sel pada tabel masih berwarna putih.
Keadaan pada saat proses:
Warna latar sel dari baris dan kolom yang dipilih pada tabel diberi berwarna abu-abu.
Keadaan setelah proses selesai: Warna latar dari setiap sel pada tabel diberi berwarna putih.

Gambar 1 Ilustrasi Penerapan Teknik Animasi Pergantian Warna pada Tabel

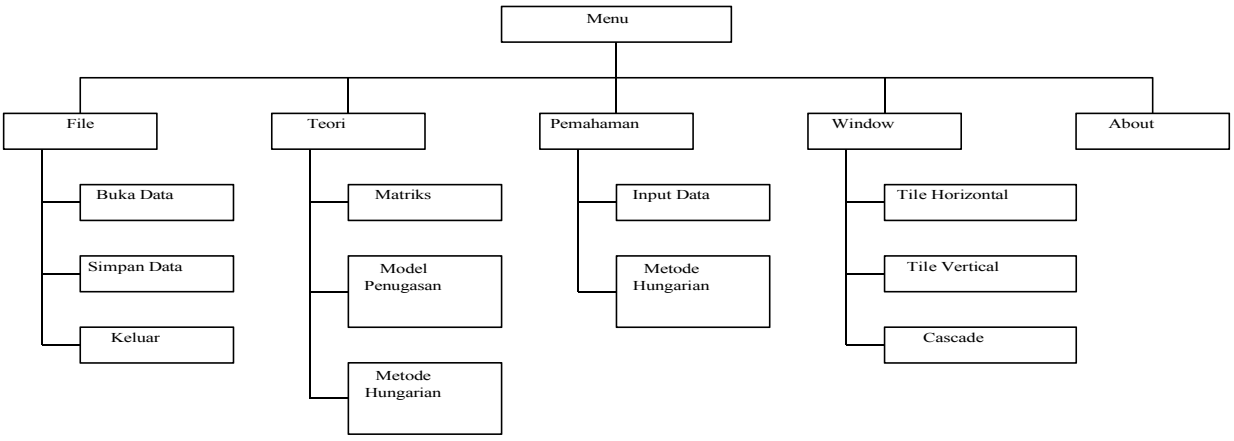
Teknik animasi lainnya yang digunakan dalam perangkat lunak pemahaman ini adalah teknik animasi pergantian gambar. Teknik ini digunakan untuk menampilkan halaman (*slide*) teori. Setiap halaman teori merupakan satu buah gambar berekstensi *.gif yang dirancang oleh penulis dan proses untuk menampilkan halaman teori adalah sama dengan proses pergantian dari gambar teori tersebut.

c. Perancangan Menu

Menu-menu yang terdapat pada *form* ini antara lain:

1. Menu '*File*', yang berfungsi untuk melakukan pengaturan proses pembukaan dan penyimpanan *file*.
2. Menu 'Teori', yang berfungsi untuk menampilkan teori-teori yang berhubungan dengan metode Hungaria
3. Menu 'Pemahaman', yang berfungsi untuk melakukan proses pemahaman mengenai cara kerja dari metode Hungaria.
4. Menu '*Window*', yang berfungsi untuk melakukan pengaturan tampilan *form*.
5. Menu '*About*', yang berfungsi untuk menampilkan data-data pribadi dari pembuat perangkat lunak.

Secara ringkas, menu-menu yang terdapat pada *form* '*Main*' ini dapat digambarkan seperti gambar 2.



Gambar 2 Rancangan Menu pada Perangkat Lunak

d. Perancangan Tampilan

Rancangan tampilan dari perangkat lunak pembelajaran penugasan pekerja dengan metode *Hungarian* ini dapat dirincikan sebagai berikut:

1. Form Main

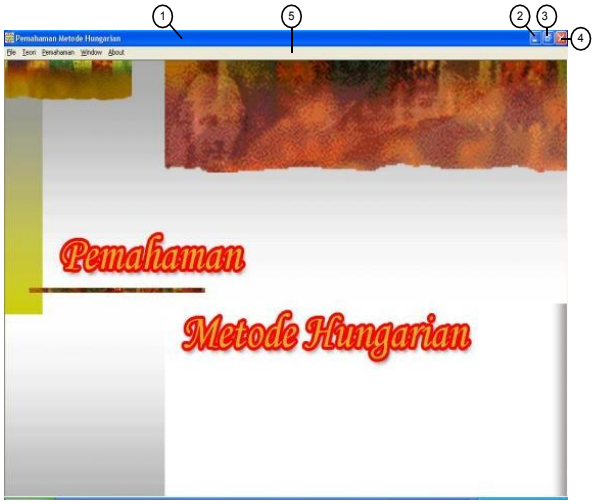
Form ini merupakan *form* inti dari perangkat lunak dan termasuk jenis *form Multiple Document Interface* (MDI). Pada *form* ini terdapat menu-menu yang digunakan sebagai penghubung (*link*) ke *form-form* lainnya.

4 : Tombol 'X', berfungsi untuk keluar dari perangkat lunak.

5 : *Menu bar*, berisi menu-menu dari perangkat lunak yang digunakan sebagai penghubung (*link*) ke *form-form* lainnya.

6 : Daerah tampilan *child form*.

2. Form Teori



Gambar 3 Rancangan Form Main

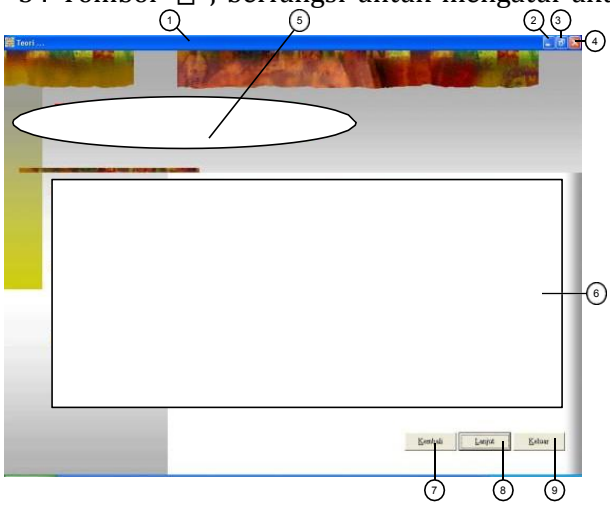
Form ini berfungsi untuk menampilkan teori-teori yang berhubungan dengan metode Hungarian.

Keterangan:

1 : *Title bar* yang berisi tulisan 'Pemahaman Metode Hungarian'.

2 : Tombol '_', berfungsi untuk me-*minimize form*.

3 : Tombol '🗗', berfungsi untuk mengatur ukuran *form*.



Gambar 4 Rancangan Form Teori

Keterangan:

1 : *Title bar* yang berisi tulisan 'Teori ...'.

2 : Tombol '_', berfungsi untuk me-*minimize form*.

3 : Tombol '🗗', berfungsi untuk mengatur ukuran *form*.
4 : Tombol 'X', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'.
5 : Daerah tampilan judul teori.
6 : Daerah tampilan teori-teori.
7 : Tombol 'Kembali', berfungsi untuk menampilkan halaman teori sebelumnya.
8 : Tombol 'Lanjut', berfungsi untuk menampilkan halaman teori selanjutnya. 9 : Tombol 'Keluar', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'.

3. Form Input

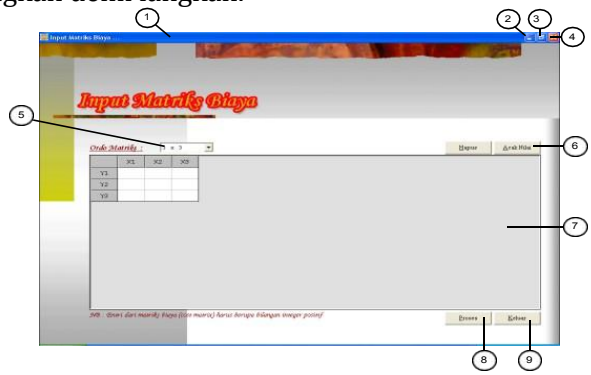
Form ini berfungsi sebagai tempat penginputan data matriks biaya.

Gambar 5 Rancangan Form Input

Keterangan:
1 : *Title bar* yang berisi tulisan 'Input Matriks Biaya ...'.
2 : Tombol '_', berfungsi untuk me-*minimize form*.
3 : Tombol '🗗', berfungsi untuk mengatur ukuran *form*.
4 : Tombol 'X', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'.
5 : *Combobox* 'Ordo', berfungsi untuk memilih ordo dari matriks biaya.
6 : Tombol 'Acak Nilai', berfungsi untuk mengisi data matriks biaya secara acak.
7 : Tabel 'Matriks Biaya', berfungsi sebagai tempat penginputan data matriks biaya.
8 8 : Tombol 'Proses', berfungsi untuk menyimpan data matriks biaya ke dalam memori sekaligus menutup *form* dan kembali ke *form* '*Main*'.
9 : Tombol 'Keluar', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'. Data matriks biaya tidak disimpan ke dalam memori.

4. Form Proses

Form ini berfungsi untuk menampilkan proses kerja dari metode Hungaria secara langkah demi langkah.

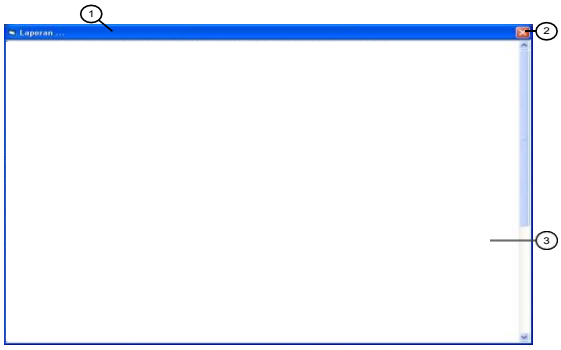


Gambar 6 Rancangan Form Proses

Keterangan:
1 : *Title bar* yang berisi tulisan 'Pemahaman Metode Hungarian ...'.

2 : Tombol '_', berfungsi untuk me-*minimize form*.
3 : Tombol '🗗', berfungsi untuk mengatur ukuran *form*.
4 : Tombol 'X', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'.
5 : Daerah tampilan keterangan mengenai langkah yang sedang dilakukan.
6 : Tabel 'Matriks Sebelumnya', berfungsi untuk menampilkan data matriks biaya sebelum dilakukan proses perhitungan.
7 : Tabel 'Matriks Sekarang', berfungsi untuk menampilkan data matriks biaya setelah dilakukan proses perhitungan.
8 : Daerah tampilan proses perhitungan yang dilakukan.
9 : Tombol 'Laporan', untuk menampilkan laporan hasil proses perhitungan.
10: Tombol 'Play', untuk menampilkan langkah perhitungan selanjutnya.
11: Tombol 'Kembali', untuk menampilkan langkah perhitungan sebelumnya.
12: Tombol 'Keluar', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'.

5. Form Laporan
Form ini berfungsi untuk menampilkan laporan hasil proses perhitungan.



Gambar 7 Rancangan Form Laporan

Keterangan:
1 : *Title bar* yang berisi tulisan 'Laporan ...'.
2 : Tombol 'X', berfungsi untuk menutup *form* dan kembali ke *form* '*Main*'.
3 : Daerah tampilan hasil proses perhitungan.

4. HASIL DAN PEMBAHASAN

4.1. ALGORITMA

Algoritma yang digunakan untuk merancang perangkat lunak pembelajaran penugasan pekerja dengan metode *Hungarian* ini dapat dibagi menjadi 3 bagian besar yaitu:

1. Algoritma Fungsi Penarikan Garis Penutup Nol

Algoritma fungsi penarikan garis penutup nol ini digunakan untuk menarik garis-garis yang melalui baris-baris dan kolom-kolom untuk menutupi entri-entri nol yang terdapat dalam matriks biaya. Fungsi ini mempunyai beberapa buah parameter *input* yaitu:

1. Variabel pnN, yang berisi nilai besar ordo matriks biaya.
2. Variabel *array* dua dimensi pArrMatrix, yang berisi nilai elemen-elemen dari matriks biaya.

Sedangkan, *output* dari fungsi ini terdiri dari:

1. Jumlah garis yang digunakan.
2. Deretan *string* yang berisi kumpulan indeks dari baris-baris yang ditutupi garis.
3. Deretan *string* yang berisi kumpulan indeks dari kolom-kolom yang ditutupi garis.

Prosedur kerja dari fungsi ini dapat dijabarkan sebagai berikut:

```
Fungsi FCrossZero(pnN, pArrMatrix)
  ArrMatrix ← pArrMatrix
  'Buang semua arsiran sebelumnya Untuk nI ← 1 sampai pnN
Untuk nJ ← 1 sampai pnN ArrMatrix(nI, nJ).bLine ← False ArrMatrix(nI, nJ).nLine ← 0
```

```
   'Cari jumlah nol terbanyak yang terdapat dalam satu baris
   nBesar ← 0
   Untuk nI ← 1 sampai pnN ArrRow(nI) ← 0
      Untuk nJ ← 1 sampai pnN
         Jika ArrMatrix(nI, nJ).nValue ← 0 And Not ArrMatrix(nI, nJ).bLine maka ArrRow(nI) ←
ArrRow(nI) + 1
            Jika ArrRow(nI) > nBesar maka nBesar ← ArrRow(nI)
   'Cari jumlah nol terbanyak yang terdapat dalam satu kolom
   Untuk nI ← 1 sampai pnN ArrCol(nI) ← 0
      Untuk nJ ← 1 sampai pnN
         Jika ArrMatrix(nJ, nI).nValue ← 0 And Not ArrMatrix(nJ, nI).bLine maka ArrCol(nI)
← ArrCol(nI) + 1
            Jika ArrCol(nI) > nBesar maka nBesar ← ArrCol(nI)
   'Cari baris yang memiliki jumlah nol sebanyak nBesar
   nJlhRow ← 0
   Untuk nI ← 1 sampai pnN
      Jika ArrRow(nI) ← nBesar maka nJlhRow ← nJlhRow + 1
   'Cari kolom yang memiliki jumlah nol sebanyak nBesar
   nJlhCol ← 0
   Untuk nI ← 1 sampai pnN
      Jika ArrCol(nI) ← nBesar maka nJlhCol
← nJlhCol + 1
   'Check apakah harus arsir secara baris atau kolom
   Jika nJlhRow >= nJlhCol maka bRow ← True
                        bCol ← False Jika tidak, maka bCol ← True
      bRow ← False End Jika
   'Loop hingga semua nol terarsir Selama nBesar > 0, lakukan proses
berikut
      Jika bRow maka
         'Arsir baris yang memiliki jumlah nol paling banyak
         Untuk nI ← 1 sampai pnN 'Check baris
            Jika ArrRow(nI) ← nBesar maka

Untuk nJ ← 1 sampai pnN ArrMatrix(nI, nJ).bLine ← True ArrMatrix(nI, nJ).nLine ←
ArrMatrix(nI, nJ).nLine + 1
               'Inkremen jumlah garis CZ.nCount ← CZ.nCount + 1 'Simpan indeks baris  CZ.cRow
               ← CZ.cRow + ";" + nI
            End Jika Jika tidak, maka
         'Arsir kolom yang memiliki jumlah nol paling banyak
         Untuk nI ← 1 sampai pnN 'Check kolom
            Jika ArrCol(nI) ← nBesar maka Untuk nJ ← 1 sampai pnN
                  ArrMatrix(nJ, nI).bLine ← True ArrMatrix(nJ, nI).nLine ←
ArrMatrix(nJ, nI).nLine + 1
               'Inkremen jumlah garis CZ.nCount ← CZ.nCount + 1 'Simpan indeks kolom CZ.cCol
               ← CZ.cCol + ";" + nI
            End Jika End Jika
      'Cari jumlah nol (yang belum diarsir) terbanyak yang
  'terdapat dalam satu baris nBesar ← 0
      Untuk nI ← 1 sampai pnN ArrRow(nI) ← 0
         Untuk nJ ← 1 sampai pnN
            Jika ArrMatrix(nI, nJ).nValue ← 0
```

```
And Not
ArrMatrix(nI, nJ).bLine maka ArrRow(nI) ← ArrRow(nI) + 1
        Jika ArrRow(nI) > nBesar maka nBesar ← ArrRow(nI)
      'Cari jumlah nol (yang belum diarsir) terbanyak yang
  'terdapat dalam satu kolom Untuk nI ← 1 sampai pnN
        ArrCol(nI) ← 0
        Untuk nJ ← 1 sampai pnN
          Jika ArrMatrix(nJ, nI).nValue ← 0
And Not
ArrMatrix(nJ, nI).bLine maka ArrCol(nI) = ArrCol(nI) + 1
        Next nJ
         Jika ArrCol(nI) > nBesar maka nBesar = ArrCol(nI)
       'Cari baris yang memiliki jumlah nol sebanyak nBesar
      nJlhRow ← 0
Untuk nI ← 1 sampai pnN
        Jika ArrRow(nI) ← nBesar maka nJlhRow ← nJlhRow + 1
      'Cari kolom yang memiliki jumlah nol sebanyak nBesar
      nJlhCol ← 0
      Untuk nI ← 1 sampai pnN
        Jika ArrCol(nI) ← nBesar maka nJlhCol ← nJlhCol + 1
      'Check apakah harus arsir secara baris atau kolom
      Jika nJlhRow >= nJlhCol maka bRow ← True
                    bCol ← False Jika tidak, maka bCol ← True
        bRow ← False End Jika
   pArrMatrix ← ArrMatrix FCrossZero ← CZ
End Fungsi
```

2. Algoritma Fungsi Pencarian Solusi

Algoritma ini digunakan untuk mencari semua kemungkinan kombinasi solusi dari matriks biaya yang dihasilkan. Fungsi ini membutuhkan beberapa buah *input* data yaitu:

1. Variabel pnN, yang berisi nilai besar ordo matriks biaya.
2. Variabel *array* dua dimensi pArrMatrix, yang berisi nilai elemen-elemen dari matriks biaya.

Sedangkan, *output* dari fungsi ini berupa deretan string yang berisi semua kemungkinan solusi. Prosedur kerja dari fungsi ini dapat dijabarkan sebagai berikut: Fungsi GetSolution(pnN, pArrMatrix)

```
  ArrMatrix ← pArrMatrix
  'Gabungkan indeks kolom yang memiliki nilai nol secara per baris
  Untuk nI ← 1 sampai pnN Untuk nJ ← 1 sampai pnN
       Jika ArrMatrix(nI, nJ).nValue ← 0
maka
ArrB(nI) ← ArrB(nI) + "," + nJ
  'Ambil semua kombinasi yang mungkin cTemp = Pecahkan ArrB(1) berdasarkan
karakter ","
  Untuk nI ← 0 sampai Indeks array tertinggi dari cTemp
    ArrSolution(nI + 1) ← cTemp(nI) nJ ← 1
  Selama nJ < pnN, lakukan proses berikut
    'Lanjut ke baris selanjutnya nJ ← nJ + 1
    'Pecahkan indeks kolom dari baris cTemp1 ← Pecahkan ArrB(nJ)
berdasarkan karakter "," 'Simpan solusi sebelumnya TArrSolution ← ArrSolution
    'Hitung jumlah hasil sebelumnya
    nJlh ← Indeks array tertinggi dari ArrSolution
```

```
    'Gabungkan dengan hasil sebelumnya Untuk nK ← 0 sampai Indeks array
tertinggi dari cTemp1
      Untuk nL ← 1 sampai nJlh ArrSolution(nL + nJlh * nK) ←
TArrSolution(nL)+ "," + cTemp1(nK) 'Checking & ambil solusi yang valid saja Untuk nI ← 1
  sampai Indeks array
tertinggi dari ArrSolution bSame ← False
    nJ ← 0
    cTemp1 ← Pecahkan ArrSolution(nI) berdasarkan karakter ","
    Selama nJ <= Indeks array tertinggi dari cTemp1 And Not
        bSame, lakukan proses berikut nK ← 0
      Selama nK <= Indeks array tertinggi dari cTemp1 And Not
          bSame, lakukan proses berikut Jika cTemp1(nJ) ← cTemp1(nK)
And nJ <> nK maka bSame ← True
        nK ← nK + 1 nJ ← nJ + 1
    Jika Not bSame maka
      cSol ← cSol + ";" + ArrSolution(nI) End Jika
  GetSolution ← cSol End Function
```

3. Algoritma Fungsi Perhitungan Metode Hungarian

Algoritma ini digunakan untuk melakukan proses pencarian solusi masalah penugasan dengan menggunakan metode Hungaria. Fungsi ini membutuhkan *input* data berupa variabel *array* dua dimensi MatrixS, yang berisi nilai elemen-elemen dari matriks biaya.

Sedangkan, *output* dari fungsi ini berupa variabel *array* 1 dimensi yang berisi data keterangan setiap langkah yang dilakukan dan proses perhitungan yang dilakukan serta perubahan nilai elemen dari matriks biaya. Prosedur kerja dari fungsi ini dapat dijabarkan sebagai berikut:

```
  '1. Ambil entri terkecil dari setiap baris & kurangkan dengan
  ' setiap entri baris tersebut Untuk nI ← 1 sampai nN
    nKecil(nI).nValue ← 100000 'Ambil entri terkecil
    Untuk nJ ← 1 sampai nN
      Jika MatrixS(nI, nJ).nValue < nKecil(nI).nValue maka
        nKecil(nI).nValue ← MatrixS(nI, nJ).nValue
        nKecil(nI).nIndex ← nJ End Jika
    'Kurangkan dengan semua entri baris Untuk nJ ← 1 sampai nN
      MatrixS(nI, nJ).nValue ← MatrixS(nI, nJ).nValue –
nKecil(nI).nValue
  '2. Ambil entri terkecil dari setiap kolom & kurangkan dengan
  ' setiap entri kolom tersebut Untuk nJ ← 1 sampai nN
    nKecil(nJ).nValue ← 100000 'Ambil entri terkecil
    Untuk nI ← 1 sampai nN
      Jika MatrixS(nI, nJ).nValue < nKecil(nJ).nValue maka
        nKecil(nJ).nValue ← MatrixS(nI, nJ).nValue
        nKecil(nJ).nIndex ← nI End Jika
    'Kurangkan dengan semua entri kolom Untuk nI = 1 sampai nN
      MatrixS(nI, nJ).nValue ← MatrixS(nI, nJ).nValue –
nKecil(nJ).nValue nStep = 2
  Selama Not bFinish, lakukan proses berikut
    nStep ← nStep + 1
    '3. Tarik garis u/ menutupi entri pada baris & kolom dengan
        ' jlh garis minimum
    TarikGrs = FCrossZero(nN, MatrixS) nStep = nStep + 1
```

```
'4. Check jlh garis, jika = N maka selesai, jika tidak maka
      ' lanjutkan proses
Jika TarikGrs.nCount = nN maka 'Proses selesai
  bFinish = True
Jika tidak, maka nStep = nStep + 1
  '5. Cari entri terkecil yang tidak tertutupi garis &
  ' kurangkan dengan semua entri yang tidak tertutupi
    ' garis & tambahkan dengan entri yang tertutupi
  ' garis dua kali
  'Cari entri terkecil yang tidak tertutupi
```

Tampilan dari perangkat lunak ini adalah sebagai berikut :

1. Tampilan *form* '*Input*' garis

```
nSmall ← 100000 Untuk nI ←
1 sampai nN
  Untuk nJ ← 1 sampai nN
    Jika MatrixS(nI, nJ).nValue <
```

```
nSmall And Not
        MatrixS(nI, nJ).bLine maka nSmall ← MatrixS(nI, nJ).nValue
          End Jika Next nI
       Untuk nI ← 1 sampai nN Untuk nJ ← 1 sampai nN
          'Kurangkan dengan semua entri yang tidak
        'tertutupi garis
          Jika MatrixS(nI, nJ).bLine ← False
maka
        MatrixS(nI, nJ).nValue ←
MatrixS(nI, nJ).nValue - nSmall End Jika
       Untuk nI ← 1 sampai nN Untuk nJ ← 1 sampai nN
          'Tambahkan dgn entri yg tertutupi garis dua kali
          Jika MatrixS(nI, nJ).nLine ← 2
```

Gambar 8 Tampilan form Input

Form '*Input*' berfungsi sebagai tempat penginputan data matriks biaya yang akan digunakan dalam perhitungan metode Hungarian. *Form* ini dapat diakses dari menu 'Pemahaman', pada submenu '*Input Data*'. Nilai elemen yang terdapat dalam matriks biaya harus berupa bilangan bulat positif dengan batasan maksimal 5 digit.

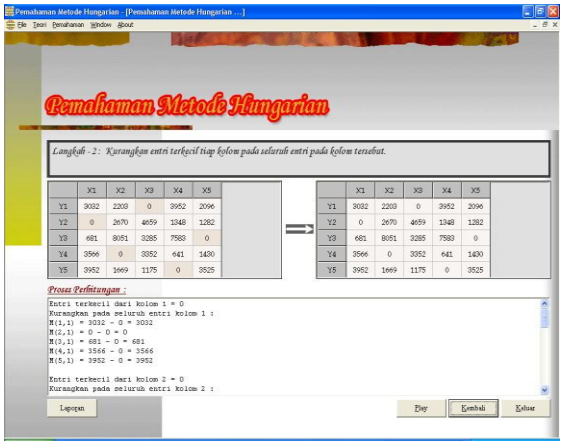
2. Tampilan *form* 'Pemahaman'

```
maka
          MatrixS(nI, nJ).nValue ←
MatrixS(nI, nJ).nValue + nSmall
          End Jika
        'Kembali ke langkah 3 bFinish ← False
      End Jika
   '6. Cari semua solusi yang mungkin nStep ← nStep + 1
   cHasil ← GetSolution(nN, MatrixS)
```

4.2 Implementasi Sistem

Implementasi sistem dari perangkat lunak ini mencakup tampilan *output* perangkat lunak dan hasil proses perhitungan dari perangkat lunak

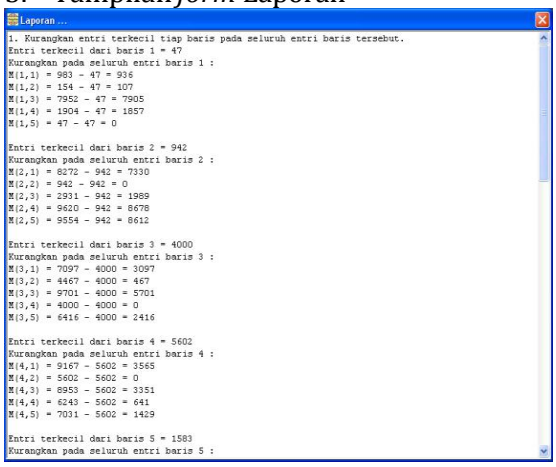
4.2.1 Tampilan Perangkat Lunak



Gambar 9 Tampilan form Pemahaman

Form 'Pemahaman' berfungsi untuk menampilkan proses perhitungan dari metode Hungarian secara langkah demi langkah. Proses perhitungan ini akan dimulai dengan menekan tombol '*Play*'.

3. Tampilan *form* Laporan



Gambar 10 Tampilan form Laporan

Form 'Laporan' berfungsi untuk menampilkan hasil proses perhitungan dari metode Hungarian. *Form* ini dapat diakses dari tombol 'Laporan' yang terdapat pada *form* 'Pemahaman'.

5. KESIMPULAN DAN SARAN

Setelah menyelesaikan tugas akhir ini, penulis menarik beberapa kesimpulan sebagai berikut : Perangkat lunak dapat digunakan untuk membantu pemahaman proses kerja dari metode Hungaria, karena perangkat lunak mampu menampilkan proses perhitungan langkah demi langkah secara terperinci. Penyelesaian model penugasan dengan menggunakan metode Hungaria dapat menghasilkan satu atau lebih solusi. Namun, juga ada beberapa permasalahan model penugasan yang tidak memiliki solusi. Jika terdapat dua atau lebih elemen matriks biaya bernilai sama, maka model penugasan tersebut akan memiliki banyak solusi, namun jika tidak, maka model penugasan tersebut akan memiliki satu solusi ataupun tidak memiliki solusi. Model penugasan tidak memiliki solusi jika dan hanya jika minimal ada elemen pada satu baris atau kolom yang tidak dapat dibuat menjadi nol.

Penulis memberikan beberapa saran, yaitu: Perangkat lunak dapat ditambahkan fasilitas suara (*multimedia*) agar proses pemahaman menjadi lebih menarik, Perangkat lunak dapat ditambahkan keterangan keterangan tambahan mengenai variabel x dan y, sehingga perangkat lunak dapat menghasilkan solusi dengan penjelasan

yang lebih terperinci.

DAFTAR PUSTAKA

Hariyanto.B, 2019., Struktur Data : Memuat Dasar Pengembangan Orientasi Objek, Edisi Kedua, Informatika Bandung,

Munir.R.R., Lidia.L. 2020, Algoritma dan Pemrograman, Edisi Kedua,.

Pramono.D. 2020, Mudah menguasai *Visual Basic* , PT. Elex Media Komputindo,

Putra.R. 2020, *The Best Source Code Visual Basic*, PT. Elex Media Komputindo,

Rorres.A. 2019, Aljabar Linear Elementer, Edisi Kedelapan – Jilid 2, Versi Aplikasi, Penerbit Erlangga,

Suryokusumo.A. 2018, *Microsoft Visual Basic* , PT. Elex Media Komputindo,

Harlod Kuhn, 2010, Algoritma Metode Hungarian, Penerbit Udik Jatmiko,