PERBANDINGAN HASIL PENGGUNAAN METODE WP DAN TOPSIS PADA SISTEM PENDUKUNG KEPUTUSAN PEMILIHAN LOKASI LAHAN TAMBAK PALING TERBAIK UNTUK DIJADIKAN USAHA TAMBAK AIR PAYAU

Yani Maulita, Relita Buaton

Program Studi Sistem Informasi, STMIK Kaputama, Binjai, 20714, Indonesia
yassa_26@ymail.com

Abstract

The number of methods available to the decision support system, so sometimes confused to choose which one matches the use of the method according to the case of a decision support system. For that made a comparison of the case of a decision support system of pond site selection to be the best business brackish water ponds for the comparison of the decision. The method used is weighted product (WP) and TOPSIS by determining the number of criteria, the type of criteria (Cost and Benefits), with three alternatives. Results of the study is the result of manual calculation equal to the calculations in the system. Each of the calculation of the WP and TOPSIS methods show that the results in the selection of the best locations embankment land to be used as business brackish water ponds each method has different outcomes depending.

Keywords: Decision Support System, WP, Topsi, Land Of Pond

1. PENDAHULUAN

Sistem pendukung keputusan adalah salah satu alteranatif menggantikan fungsi manajemen untuk mengotomatisasikan pengambilan keputusan dengan untuk melakukan berbagai analisis menggunakan model-model yang tersedia. Banyaknya metode-metode yang tersedia pada sistem pendukung keputusan sehingga kadang bingung memilih mana yang cocok penggunaaan metode yang sesuai dengan kasus sistem pendukung keputusan. WP dan Topsis adalah salah satu metode atau model multi attribut decision making yang paling banyak digunakan untuk sistem pendukung keputusan. Untuk itu dibuat suatu perbandingan hasil keputusan dari kasus sistem pendukung keputusan pemilihan lokasi lahan tambak paling terbaik untuk dijadikan usaha tambak air. masalah dalam penelitian ini yaitu sebagai berikut bagaimana sistem dapat memproses data sesuai dengan spesifikasi pemilihan lokasi lahan tambak yang paling terbaik untuk dijadikan usaha tambak air? dan bagaimana penggunaan metode yang digunakan pada sistem dapat memberikan hasil perbandingan sesuai dengan kriteria-kriteria yang ada? Berdasarkan identifikasi dapat masalah dapat dirumuskan yaitu Dengan sistem pendukung keputusan, bagaimana memproses data sesuai dengan spesifikasi pemilihan lokasi lahan tambak yang paling terbaik untuk dijadikan usaha tambak air dan dengan penggunaan metode WP dan Topsis, bagaimana sistem dapat memberikan hasil perbandingan dari kedua metode menampilkan proses tahapan (algoritma) pertama sampai ke perangkingan alternatif beserta keterangan dari tiap tahapan yang ada. Tujuan dari penelitian ini adalah Membuat sistem pendukung keputusan pemilihan lokasi lahan tambak paling terbaik untuk dijadikan usaha tambak air payau serta mengetahui hasil perbandingan dari metode WP dan Topsis sebagai alternatif analisa untuk pengambilan keputusan. Diharapkan manfaat dari tulisan ini adalah dapat membantu pihak dinas terkait dalam pengambilan keputusan pemilihan lokasi lahan tambak paling terbaik untuk dijadikan usaha tambak air. kemudian manfaat lainnnya adalah dapat membantu pihak dinas terkait dalam memberikan alternatif hasil alternatif dari perbandingan metode wp dan topsis

2. TEORI

***Decision Support System* (DSS)**

Menurut Alter (Kusrini, 2007, h. 15-16) Sistem Pendukung Keputusan (SPK) atau Decision Support System (DSS) adalah sistem informasi interaktif yang menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan dalam situasi yang semi terstruktur dan situasi yang tidak terstruktur, dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. SPK ditujukan untuk mendukung manajemen dalam melakukan pekerjaan yang bersifat analitis dalam situasi yang kurang terstruktur dengan kriteria yang kurang jelas. DSS tidak dimaksudkan untuk mengotomatiskan pengambilan keputusan, tetapi memberikan perangkat interaktif yang memungkinkan pengambil keputusan untuk melakukan berbagai analisis menggunakan model-model yang tersedia.

Keputusan merupakan kegiatan memilih suatu strategi atau tindakan dalam pemecahan masalah. Tujuan dari keputusan adalah untuk mencapai target atau aksi tertentu yang harus dilakukan. Kriteria atau ciri-ciri dari keputusan adalah (Kusrini, 2007, h.7) :

1. Banyak pilihan / alternatif.
2. Ada kendala atau syarat.
3. Mengikuti suatu pola/model tingkah laku, baik yang terstruktur maupun tidak terstruktur.
4. Banyak *input*/variabel.
5. Ada faktor resiko.
6. Dibutuhkan kecepatan, ketepatan dan keakuratan.

Keputusan yang diambil untuk menyelesaikan suatu masalah dilihat dari keterstukturannya yang bisa dibagi menjadi (Kusrini, 2007, h. 19-20) :

1. Keputusan terstruktur (*structured decision*)
   Keputusan terstruktur merupakan keputusan yang dilakukan secara berulang-ulang dan bersifat rutin. Keputusan ini terutama dilakukan pada manajemen tingkat bawah. Misalnya, keputusan pemesanan barang dan keputusan penagihan piutang.
2. Keputusan semiterstruktur (*semistructured decision*)
   Keputusan semiterstruktur merupakan keputusan yang memiliki dua sifat. Sebagian

keputusan bisa ditangani oleh komputer dan yang lain tetap harus dilakukan oleh pengambil keputusan. Prosedur dalam pengambilan keputusan tersebut secara garis besar sudah ada, tetapi ada beberapa hal yang masih memerlukan kebijakan dari pengambil keputusan. Keputusan ini diambil oleh manajer level menengah dalam suatu organisasi. Misalnya, pengevaluasian kredit, penjadwalan produksi dan pengendalian sediaan.

3. Keputusan tidak terstruktur (*unstructured decission*)
   Keputusan tidak terstruktur merupakan keputusan yang penanganannya rumit karena tidak terjadi berulang-ulang atau tidak selalu terjadi. Keputusan ini menuntut pengalaman dan berbagai sumber yang bersifat eksternal. Keputusan ini umumnya terjadi pada manajemen tingkat atas. Misalnya, keputusan pengembangan teknologi baru, keputusan untuk bergabung dengan perusahaan lain, perekrutan eksekutif.

Saat melakukan pemodelan dalam pembangunan DSS dilakukan langkah-langkah sebagai berikut (Kusrini, 2007, h. 30-31) :

1. Studi kelayakan (*Intelligence*)
   Pada langkah ini, sasaran ditentukan dan dilakukan pencarian prosedur, pengumpulan data, identifikasi masalah, identifikasi kepemilikan masalah, klasifikasi masalah, hingga akhirnya terbentuk sebuah pernyataan masalah, Kepemilikan masalah berkaitan dengan bagian apa yang akan dibangun oleh DSS dan apa tugas dari bagian tersebut sehingga model tersebut bisa relevan dengan kebutuhan si pemilik masalah.
2. Perancangan (*Design*)
   Pada tahapan ini akn diformulasikan model yang akan digunakan dan kriteria-kriteria yang ditentukan. Setelah itu, dicari alternatif model yang bisa menyelesaikan permasalahan tersebut. Langkah selanjutnya adalah memprediksi keluaran yang mungkin. Kemudian ditentukan variabel-variabel model.
3. Pemilihan (*Choise*)
   Setelah pada tahap *design* ditentukan berbagai alternatif model beserta variabel-variabelnya, pada tahapan ini akan dilakukan pemilihan modelnya, termasuk solusi dari model tersebut. Selanjutnya dilakukan analisis sensitivitas, yakni dengan mengganti beberapa variabel.
4. Membuat DSS
   Setelah menentukan modelnya, berikutnya adalah mengimplementasikannya dalam aplikasi DSS.

***Multi-Attribute Decision Making* (MADM)**

Menurut Kusumadewi, dkk (2006, h. 74) beberapa model atau metode yang dapat digunakan untuk menyelesaikan masalah MADM yaitu :

1. *Simple Additive Weighting* (SAW).
2. *Weighted Product* (WP).
3. *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS).
4. *Analytic Hierarchy Process* (AHP).
5. *Elemination Et Choix TRaduisant la realitE* (ELECTRE).

Weighted Product (WP)

Menurut Yoon (Kusumadewi, dkk, 2006, h. 79) Metode WP menggunakan perkalian untuk menghubungkan *rating* atribut, dimana *rating* setiap atribut harus dipangkatkan dulu dengan bobot atribut yang bersangkutan. Proses ini sama halnya dengan proses normalisasi. Sebelumnya akan dilakukan perbaaikan bobot dengan cara :

$$w_j = \frac{w_j}{\sum w_j}$$

Preferensi untuk alternatif A_i diberikan sebagai berikut :

$$S_i = \prod_{j=1}^{n} x_{ij}^{wj} \quad ; \text{dengan } i=1,2,\ldots m.$$

Dimana $\sum w_j = 1$. w_j adalah pangkat bernilai positif untuk atribut keuntungan, dan bernilai negatif untuk atribut biaya.

Preferensi relatif dari setiap alternatif, diberikan sebagai :

$$V_i = \frac{\prod_{j=1}^{n} x_{ij}^{wj}}{\prod_{j=1}^{n} (x_j^*)^{wj}} \quad ; \text{dengan } i = 1,2,\ldots m.$$

***Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)**

Menurut Hwang dan Zeleny (Kusumadewi, dkk, 2006, h. 87) TOPSIS didasarkan pada konsep dimana alternatif terpilih yang terbaik tidak hanya memiliki jarak terpendek dari solusi ideal positif, namun ada juga memiliki jarak terpanjang dari solusi ideal negatif. Konsep ini banyak digunakan pada beberapa model MADM untuk menyelesaikan masalah keputusan secara praktis. Hal ini disebabkan karena konsepnya sederhana dan mudah dipahami, komputasinya efesien, dan memilki kemampuan untuk mengukur kinerja relatif dari alternatif-alternatif keputusan dalam bentuk matematis yang sederhana.

Secara umum, prosesdur TOPSIS mengikuti langkah-langkah sebagai berikut :

1. Membuat matriks keputusan yang ternormalisasi.
2. Membuat matriks keputusan yang ternormalisasi terbobot.
3. Menentukan matriks solusi ideal positif dan matriks solusi ideal negatif.
4. Menentukan jarak antara nilai setiap alternatif dengan matriks solusi ideal positif dan matriks solusi ideal nagatif.
5. Menentukan nilai preferensi untuk setiap alternatif.

TOPSIS membutuhkan rating kinerja setiap alternatif A_i pada setiap kriteria C_j yang ternormalisasi yaitu :

$$r_{ij} = \frac{x^{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \quad ; \text{dengan } i=1,2,\ldots,m. \text{ dan } j=1,2,\ldots,n.$$

Solusi Ideal Positif A^+ dan solusi ideal negatif A^- dapat ditentukan berdasarkan rating bobot ternormalisasi (y_{ij}) sebagai :

$y_{ij} = w_i r_{ij}$; dengan i=1,2,…,m. dan

j=1,2,…,n.

$A^+ = (y_1^+, y_2^+, \ldots, y_n^+)$;

$A^- = (y_1^-, y_2^-, \ldots, y_n^-)$;

Dengan

$$y_j^+ = \begin{cases} max_i y_{ij} & \text{; jika j adalah atribut keuntungan} \\ min_i y_{ij} & \text{; jika j adalah atribut biaya} \end{cases}$$

$$y_j^- = \begin{cases} min_i y_{ij} & \text{; jika j adalah atribut keuntungan} \\ max_i y_{ij} & \text{; jika j adalah atribut biaya} \end{cases}$$

j=1,2,…,n.

Jarak antara alternatif A_i dengan solusi ideal positif dirumuskan sebagai :

$$D_i^+ = \sqrt{\sum_{j=1}^{n} (y_i^+ - y_{ij})^2} \quad ; i=1,2,\ldots m.$$

Jarak antara alternatif A_i dengan solusi ideal negatif dirumuskan sebagai :

$$D_i^- = \sqrt{\sum_{j=1}^{n} (y_{ij} - y_i^-)^2} \quad ; i=1,2,\ldots m.$$

Nilai preferensi untuk setiap alternatif (V_i) diberikan sebagai :

$$V_i = \frac{D_i^-}{D_i^- + D_i^+} \quad ; i=1,2,\ldots m.$$

Nilai V_i yang lebih besar menunjukkan bahwa Alternatif A_i lebih dipilih.

3. ANALISA DAN PERANCANGAN SISTEM

Kabupaten Suna mengadakan rapat yang membahas tentang anggaran pembangunan daerah. Salah satu pejabat tinggi memberikan masukan kepada Bupati Suna untuk membuka lahan usaha tambak air payau, karena dia berpendapat desa-desa yang ada di Kabupaten Suna banyak muara. Kepala Dinas Peternakan dan Perikanan sudah lama memikirkan hal ini, dan ia memberikan masukan bahwa ada tiga dari delapan desa yang cocok dijadikan tambak air payau karena tiga lahan tersebut berjenis tanah Aluvium dengan kadar garam 0,05% - 3%. Ia menjelaskan yang harus diperhitungkan dalam membuka lahan tambak air payau yaitu kadar air, tekstur tanah dan lereng tanah. Ia juga menjelaskan bahwa ketiga lahan tersebut memiliki penilaian yang hampir sama. Bupati Suna sangat setuju dengan masukan itu dan mengusulkan bahwa akan membangun dua Tambak Air Payau dan ia merasa ini akan menjadi lapangan pekerjaan baru bagi masyarakat yang masih mengganggur di desa-desa sekitarnya. Akan tetapi, Bendahara Kabupaten keberatan jika langsung membuka dua lahan sekaligus karena anggaran sudah hampir mencapai limit. Sisa anggaran yang ada saat ini hanya dapat membuka satu lahan. Dan Bupati menyuruh Dinas Perternakan dan Perikanan untuk segera mencari satu lahan yang paling bagus untuk dijadikan usaha tambak air payau.

Berdasarkan dari kasus diatas ada tiga lokasi yang menjadi alternatif yaitu :

A1 = Desa Kabukico
A2 = Desa Duton Batu
A3 = Desa Katun

Dan ada lima kriteria penilaian dalam memilih satu lahan pengambilan keputusan:

K1 = Kadar Air / Salinitas Air (Garam Terlarut)
K2 = Tekstur Tanah
K3 = Lereng Tanah

Adapun penilaian di setiap kriteria sebagai berikut :

Tabel 1 Penilaian Kriteria

Kadar Air	Nilai
0.05% - 1%	1
2.01% - 3%	2
1.01% - 2%	3

Tekstur Tanah	Nilai
Berpasir Halus	1
Berpasir Sangat Halus	2
Liat	3

Lereng Tanah	Nilai
3%	1
2%	2
0 - 1%	3

Tingkat kepentingan yang akan dijadikan bobot preferensi setiap kriteria sebagai berikut:

Tabel 2 Bobot Kriteria

Nilai	Bobot
Kurang Penting	1
Penting	2
Sangat Penting	3

Adapun *Rating* Kecocokan dari setiap Alternatif pada setiap Kriteria sebagai berikut:

Tabel 3 *Rating* Kecocokan dari setiap Alternatif pada setiap Kriteria

Alternatif	K1	K2	K3
A1	2	2	3
A2	3	3	1
A3	3	2	2
Bobot (W)	**3**	**2**	**2**

Dengan data yang ada, maka yang akan dilakukan sebagai berikut :

1. Perbaikan Bobot
   Σ BOBOT : (3 + 2 + 2) = 7
   K1 = 3 / 7 = 0.43
   K2 = 2 / 7 = 0.29
   K3 = 2 / 7 = 0.29
2. Vektor S
   $S_1 = (2^{0.43}) \times (2^{0.29}) \times (3^{0.29}) = 2.2456$
   $S_2 = (3^{0.43}) \times (3^{0.29}) \times (1^{0.29}) = 2.1918$
   $S_3 = (3^{0.43}) \times (2^{0.29}) \times (2^{0.29}) = 2.3796$
3. Vektor V

$$V_1 = \frac{2.2456}{2.2456 + 2.1918 + 2.3796} = 0.3294$$

$$V_2 = \frac{2.1918}{2.2456 + 2.1918 + 2.3796} = 0.3215$$

$$V_3 = \frac{2.3796}{2.2456 + 2.1918 + 2.3796} = 0.3491$$

Berdasarkan hasil Vektor V diatas maka akan dilakukan perangkingan alternatif sebagai berikut:

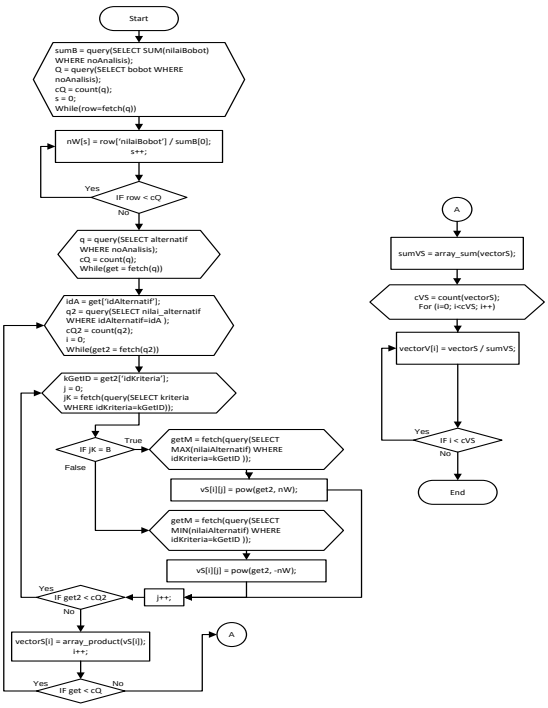
Tabel 4 Rangking Hasil Analisis Metode WP

Rangking	Alternatif	Hasil
1.	Desa Katun	0.3491
2.	Desa Kabukico	0.3294
3.	Desa Duton Batu	0.3215

Berdasarkan hasil perhitungan metode WP diatas, Desa Katun merupakan alternatif yang terbaik untuk di bangunnya lahan usaha tambak air payau dengan nilai 0.3491.

Metode WP menggunakan perkalian untuk menghubungkan rating atribut, dimana rating setiap atribut harus dipangkatkan dulu dengan bobot atribut yang bersangkutan. Adapun penjelelasan dan *flowchart* program algoritma metode WP yaitu:

1. Memanggil "nilaiBobot" untuk perbaikan bobot, dengan cara $bobot_i$ dibagi dengan jumlah bobot yang ada.
2. Memanggil "nilaiAlternatif" sesuai dengan "idAlternatif". Selanjutnya menseleksi jenis kriteria dari setiap Nilai Alternatif berdasarkan dari idKriteria dilangkah ini akan digunakan kondisi IF. Sebelumnya dilakukan pemangkatan (*pow*) dengan bobot baru disetiap nilaiAlternatif yang ada. Setalah itu nilai-nilai yang sudah dipangkatkan akan dikalikan dengan fungsi *array_product* maka akan mendapatkan vectorS.
3. Selanjutnya mencari nilai akhir (*vectorV*) dengan cara *vectorS* dibagai dengan array_sum(*vectorS*). *vectorV* akan disimpan kedalam *database*.



Gambar1 *Flowchart* Algoritma Metode WP

Berikut ini tabel Rangking Hasil Analisis Metode TOPSIS:

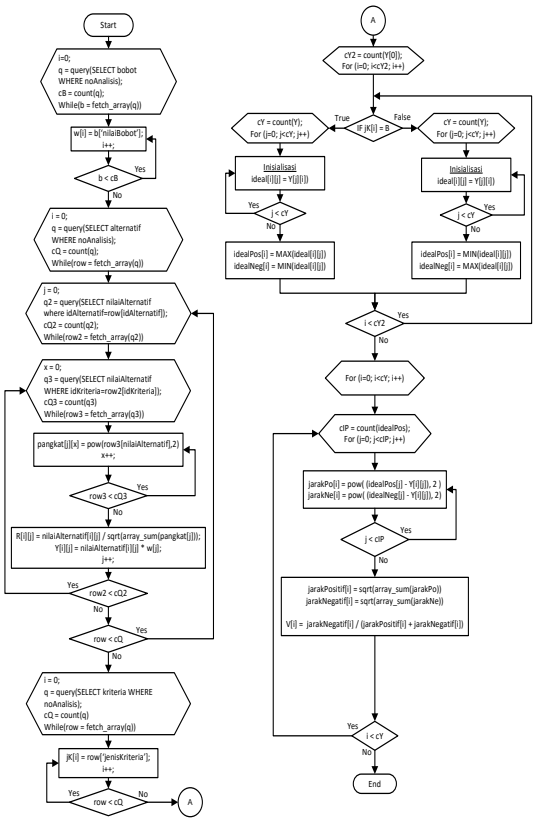
Tabel III.5 Rangking Hasil Analisis Metode TOPSIS

Rangking	Alternatif	Hasil
1.	Desa Kabukico	0.5711
2.	Desa Katun	0.5359
3.	Desa Duton Batu	0.4289

Berdasarkan dari hasil perhitungan metode TOPSIS, Desa Kabukico merupakan alternatif yang tertinggi dengan nilai 0.5711.

Adapun penjelelasan dan *flowchart* program algoritma metode TOPSIS yaitu :

1. Memanggil nilaiBobot dari *database* dan memberi inisial w[i].
2. Memanggil nilaiAlternatif berdasarkan idAlternatif dari *database* dan dilakukan pangkat[j][x] = pow(nilaiAlternatif,2) dari setiap kolom Kriteria.
3. Mencari R[i][j] = SQRT(nilaiAlternatif[i] / array_sum(pangkat[j]). SQRT merupakan fungsi akar sedangkan array_sum adalah fungsi penjumlahan nilai array. Nilai langsung di hitung dengan cara, Y[i][j] = R[i][j] * w[j].
4. Setelah mendapatkan nilai Y[i][j] maka selanjutnya akan memanggil jenisKriteria dari database.
5. Mecari solusi ideal positif dan negatif berdasarkan jenisKriteria dari setiap kolom Y. jika jenisKriteria=Benefit maka idealPos[i] adalah MAX dan idealNeg[i] adalah MIN. Sedangkan jika jenisKriteria=cost merupakan negasi dari Benefit.
6. Selanjutnya menentukan nilai jarakPositif[i] berdasarkan idealPos[i] dan jarakNegatif berdasarkan idealNeg.
7. Funtuk formula hasil akhir yaitu v[i] = jarakPositif[i] / (jarakPositif[i] + jarakNegatif[i]).
8. Nilai V akan disimpan ke dalam *database*.



Gambar.2 *Flowchart* Algoritma Metode TOPSIS

4. HASIL

Matriks Keputusan (X)

Alternatif	Kriteria		
	K1	K2	K3
A1	2	2	3
A2	3	3	1
A3	3	2	2
Bobot (*W*)	3	2	2

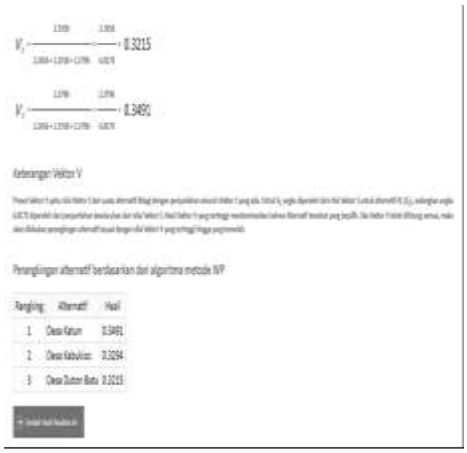
Keterangan Matriks Keputusan (X) :

Nama Kriteria

K1 = Kadar Air / Salinitas Air - Keuntungan (*Benefit*)

K2 = Tekstur Tanah - Keuntungan (*Benefit*)

Gambar 3 Data yang Telah dimasukkan dengan Metode TOPSIS

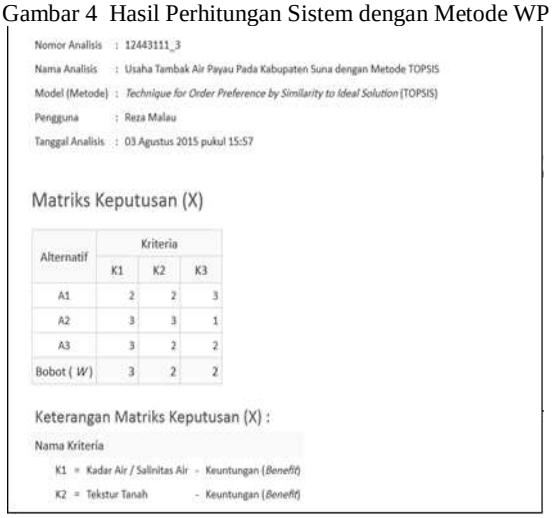


Keterangan Vektor V

Perangkingan alternatif berdasarkan dari algoritma metode WP

Rangking	Alternatif	Hasil
1	Desa Katun	0.3481
2	Desa Kabulsien	0.3294
3	Desa Dusun Batu	0.3215

Gambar 4 Hasil Perhitungan Sistem dengan Metode WP



Nomor Analisis : 12443111_3

Nama Analisis : Usaha Tambak Air Payau Pada Kabupaten Suna dengan Metode TOPSIS

Model (Metode) : *Technique for Order Preference by Similarity to Ideal Solution* (TOPSIS)

Pengguna : Reza Malau

Tanggal Analisis : 03 Agustus 2015 pukul 15:57

Matriks Keputusan (X)

Alternatif	Kriteria		
	K1	K2	K3
A1	2	2	3
A2	3	3	1
A3	3	2	2
Bobot (*W*)	3	2	2

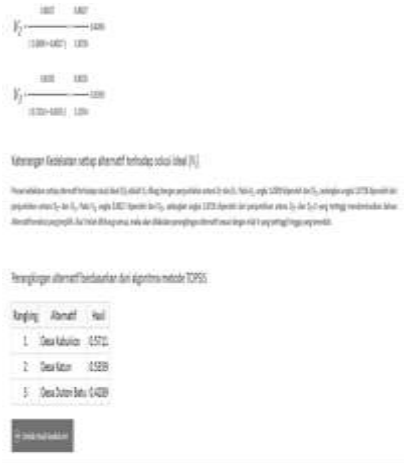
Keterangan Matriks Keputusan (X) :

Nama Kriteria

K1 = Kadar Air / Salinitas Air - Keuntungan (*Benefit*)

K2 = Tekstur Tanah - Keuntungan (*Benefit*)

Gambar 5 Data yang Telah dimasukkan dengan Metode TOPSIS



Perangkingan alternatif berdasarkan dari algoritma metode TOPSIS

Rangking	Alternatif	Hasil
1	Desa Kabulsien	0.5721
2	Desa Katun	0.5039
3	Desa Dusun Batu	0.4289

Gambar 6 Hasil Perhitungan Sistem dengan Metode TOPSIS


Laporan Hasil Perhitungan

Hasil dari perhitungan ini merupakan pendukung pemecahan masalah dalam pengambilan keputusan multi-kriteria dengan metode Weighted Product (WP). Adapun data dari "Usaha Tambak Air Payau Pada Kabupaten Buna dengan Metode WP" yang sudah sesuai dengan masukkan dari Reza Malau melalui E-Learning DSS yaitu sebagai berikut :

Alternatif	Kriteria		
	K1	K2	K3
A1	3	3	3
A2	3	3	1
A3	3	2	2
Bobot	3	2	2

Keterangan Data :

Kriteria :

K1 = Kadar Air / Salinitas Air = "Keuntungan"
K2 = Tekstur Tanah = "Keuntungan"
K3 = Lereng Tanah = "Keuntungan"

Alternatif :

A1 = Desa Kabukico
A2 = Desa Duton Batu
A3 = Desa Katun

Adapun hasil perhitungan dari data diatas, berikut adalah rangking alternatif dengan hasil tertinggi hingga yang terendah sesuai dengan perhitungan metode Weighted Product (WP) yaitu sebagai berikut :

Rangking	Alternatif	Hasil
1	Desa Katun	0.3491
2	Desa Kabukico	0.3294
3	Desa Duton Batu	0.3215

Catatan : Untuk informasi lebih lanjut seperti tahapan penyelesaian perhitungan sebagai pembelajaran atau yang lainnya, silahkan

Gambar 7 Laporan Hasil Perhitungan PDF

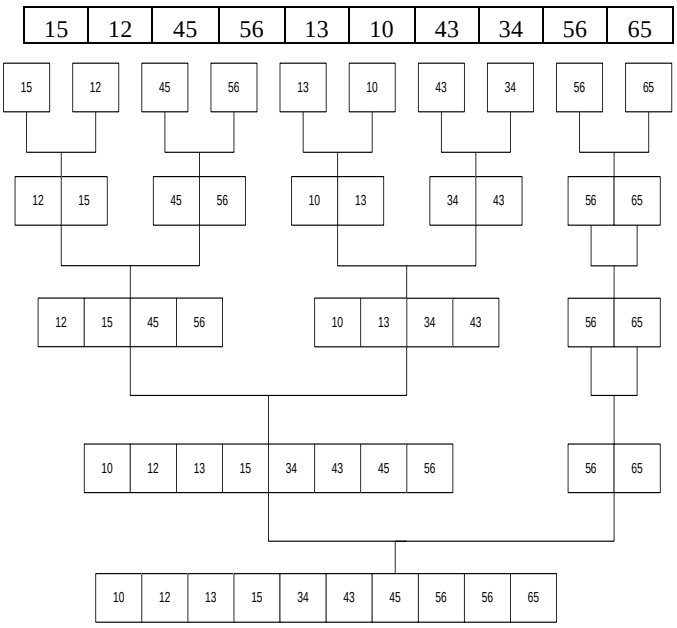
5. KESIMPULAN

Berdasarkan hasil penelitian dan pembahasan yang telah dilakuka dapat diambil beberapa kesimpulan antara lain : Dari hasil pengujian sistem yang telah dilakukan, maka dapat diberikan beberapa kesimpulan diantaranya sebagai berikut :

1. Hasil perhitungan manual sama dengan perhitungan yang ada pada sistem.
2. Berdasarkan hasil perhitungan kedua metode:
   a. Hasil perhitungan dengan metode WP yaitu: Desa Katun merupakan alternatif yang terbaik untuk di bangunnya lahan usaha tambak air payau dengan nilai 0.3491.
   b. Hasil perhitungan dengan metode TOPSIS yaitu: Desa Kabukico merupakan alternatif yang tertinggi dengan nilai 0.5711.
1. Berdasarkan dari perhitungan dari kasus sistem pendukung keputusan pemilihan lokasi lahan tambak paling terbaik untuk dijadikan usaha tambak air payau menunjukkan bahwa setiap metode WP dan Topsis memiliki hasil akhir yang berbeda-beda dan perangking untuk sebuahl keputusan.

DAFTAR PUSTAKA

[1] Adi Nugroho, *Perancangan dan Implementasi Sistem Basis Data*. Edisi pertama, Andi, Yogyakarta, 2011.
[2] Afriyudi, *Pemrograman Web Dinamis dengan Kolaborasi PHP dan Java*. Edisi Pertama, Andi, Yogyakarta, 2008.
[3] Janner Simarmata, *Rekayasa Web*. Edisi Pertama, Andi, Yogyakarta, 2010.
[4] Janner Simarmata dan Iman Paryudi. 2006. *Basis Data*. Edisi Pertama, Andi, Yogyakarta, 2006.
[5] Kusrini, *Konsep dan Aplikasi Sistem Pendukung Keputusan*. Edisi Pertama, Andi, Yogyakarta, 2007
[6] Meri Azmi, dkk, *Pemanfaatan Sistem Pendukung Keputusan untuk Penentuan Alokasi Dana Kegiatan (Studi Kasus Unit Kegiatan Mahasiswa Politeknik Negeri Padang)*. Jurnal, Vol.16, No.1, ISSN.1693-752X, h.74-83, Politeknik Negeri Padang, Padang, 2014.
[7] Sri Kusumadewi, dkk, *Fuzzy Multi-Attribute Decision Making (Fuzzy MADM)*. Edisi Pertama, Cetakan Pertama, Graha Ilmu, Yogyakarta, 2006.
[8] Stendy B. Sakur, *PHP 5 Pemrograman Berorientasi Objek – Konsep & Implementasi*. Edisi Pertama, Andi, Yogyakarta, 2010.
[9] Suarga, *Algoritma Pemrograman*. Edisi Kedua, Andi, Yogyakarta, 2012.
[10] Wahana Komputer, *Paduan Belajar MySQL Database Server*. Cetakan Pertama, Media Kita, Jakarta, 2010.



Pada prosedur yang akan disajikan di bawah ini, terdapat dua buah prosedur bantu. Prosedur pertama adalah prosedur *merge* yaitu prosedur yang digunakan untuk melakukan *merge* terhadap dua buah vektor dengan panjang tertentu.

Merge sort menggabungkan dua ide utama untuk meningkatkan *runtime*-nya:

1. *Array* kecil akan mengambil langkah-langkah untuk menyortir lebih sedikit dari *array* besar.
2. Lebih sedikit langkah yang diperlukan untuk membangun sebuah *array* terurut dari dua buah *array* terurut daripada dari dua buah *array* tak terurut.

A. *Algoritma Merge Sort*

Langkah 0 Baca elemen vektor yang akan diurutkan, misalnya vektor v, dan cacah elemen N.

Langkah 1 (menbentuk senarai berantai)
Untuk I=1 sampai N, tentukan :
Senarai[I].Info = v[I], dan
Senarai[I].kanan = I+1.
Kemudian, tentukan : senarai[N].kanan = 0.

Langkah 2 Tentukan : awal = 1 (awal senarai berantai).

Langkah 3 Kerjakan langkah sampai 16 untuk K = 1 sampai angka (angka adalah cacah digit terbanyak dari elemen vektor yang akan diurutkan).

Langkah 4 (Inisialisasi antrian).
Untuk I = 0 sampai 9, tentukan : Belakang [I] = 0.
Untuk I = 0 sampai 10, tentukan : Depan [I] = 0.

Langkah 5 (Memecahkan senarai berantai menjadi sejumlah antrian)
Kerjakan langkah 6 sampai 9 selama awal <> 0.

Langkah 6 Tentukan : P = awal,
Awal = senarai [Awal].Kanan, dan
Y = senarai [P].Info.

Langkah 7 (Menentukan digit satuan atau puluhan atau ratusan atau ribuan dan seterusnya)
Tentukan : Pangkat = 1.
Untuk I = 1 sampai K-1 :
Tentukan : Pangkat = Pangkat * 10.
Kemudian, tentukan : J = (Y div Pangkat) mod 10
(nilai digit satuan atau puluhan atau ratusan atau ribuan dan seterusnya).

Langkah 8 (Menempatkan dalam antrian)
Tentukan : Q = Belakang [J].
Test apakah : Q = 0 ?
Jika ya, tentukan : Depan [J] = P.
Jika tidak, tentukan : senarai [Q].Kanan = P.

Langkah 9 Tentukan : Belakang [J] = P.

Langkah 10 (Menyusun kembali antrian menjadi senarai berantai)
Tentukan : J = 0
Tentukan : J + 1 selama (J <= 9) dan (Depan [J]) = 0.

Langkah 11 Tentukan : Awal = Depan [J], (awal antrian).

Langkah 12 Kerjakan langkah 13 sampai 15 selama J <= 9.

Langkah 13 Tentukan : I = J + 1
Tentukan : I = I + 1 selama (I <= 9) dan (Depan [I] = 0).

Langkah 14 Test apakah I <= 9 ?
Jika ya, tentukan : P = I, dan
Senarai [Belakang [J]].Kanan = Depan [I].

Langkah 15 Tentukan : J = I.

Langkah 16 (Akhir senarai berantai)
Tentukan : senarai [Belakang [p]].Kanan = 0.

Langkah 17 (Mengambil bentuk senarai berantai menjadi vektor)
Untuk I = 1 sampai N, kerjakan :
V [I] = senarai [Awal].Info, dan
Awal = senarai [Awal].Kanan.

Langkah 18 selesai.

B. Kompleksitas Algoritma Merge Sort

Merge Sort merupakan algoritma pengurutan yang baik terutama untuk mengurutkan data yang jumlahnya sangat banyak. Untuk data yang sedikit, algoritma ini sebaiknya tidak digunakan karena ada beberapa algoritma lain yang bisa bekerja lebih cepat dari *Merge Sort*. Algoritma ini mengaplikasikan pembagian elemen array menjadi 2 buah *sub-array*. Lalu menggabungkan data yang ada pada kedua buah *sub-array* tersebut. Kompleksitas waktu untuk semua kasus dari algoritma *Merge Sort* adalah *O(n log n)*. Untuk menghitung kompleksitas dari algoritma *Merge Sort* bisa menerapkan pembagian *General* dan teorema *Conquer*.

Diketahui bahwa:

$$T(n) = aT\left(\frac{n}{b}\right) + O(n^k)$$

Dimana:
n : merupakan jumlah masalah utama
a : merupakan jumlah dari sub-masalah
n/b : merupakan jumlah pembagian data
$O(n^k)$: kompleksitas dari pembagian dan operasi *merging*

$T(n) = O(n^{logba})$ jika $a > b^k$
$T(n) = O(n^k \log n)$ jika $a = b^k$
$T(n) = O(n^k)$ jika $a < b^k$

Pada *Merge Sort*, masalah utama dibagi kedalam 2 sub-masalah, jadi a = 2. Jumlah dari masalah utama dibagi dengan 2, jadi b = 2. Kompleksitas untuk pembagian adalah *O (1)* dan kompleksitas waktu untuk *merging* adalah *O (n)*. Total waktu untuk pembagian dan *merging* adalah *O (1) + O (n) = O (n) = O (n1)*. Jadi, k = 1. Karena $a = b^k$ maka kompleksitas dari *Merge Sort* adalah *O (nklog n) = O (n log n)*.

Kasus terbaik (*Best Case*) untuk algoritma *Merge Sort* ini yaitu pada saat data inputan berupa data yang terurut baik *ascending* maupun *descending*. Sedangkan kasus terburuknya (*Worst Case*) yaitu pada saat data inputan berupa data yang acak (*random*). Tetapi untuk kompleksitas waktu asimptotiknya sama, yaitu sebesar *O (n2log n)*.

V. ANALISA ALGORITMA

A. *Metode Quick Sort*

Metode *Quick Sort* memiliki Algoritma yang lebih rumit, tetapi hasilnya lebih cepat karena hanya memerlukan sejumlah langkah yang lebih sedikit untuk menghasilkan hal yang sama. Proses pengurutan data dengan menggunakan metode *Quick Sort* secara garis besar dapat dijelaskan sebagai berikut. Apabila diketahui sebuah vektor K dengan N buah elemen data yang akan diurutkan secara urut naik. Pertama-tama dipilih salah satu elemen data sembarang pada vektor K tersebut, biasanya adalah elemen apada urutan data pertama dan diberi nama X. Selanjutnya, semua elemen pada vektor K disusun dengan menempatkan elemen data X pada posisi data tertentu yaitu J sedemikian rupa sehingga elemen data pada urutan ke-1 samapai ke j-1 mempunyai harga lebih besar dari X. Sehinggga akan terdapat dua subvektor, yaitu subvektor bagian pertama yang memuat elemen-elemen data yang memiliki harga kurang dari X dan subvektor bagaian kedua yang memuat elemen-elemen data yang memiliki harga lebih dari X.

Prosedur di atas akan diulang pada subvektor pertama dan subvektor kedua, sehingga akan diperoleh emapat subvektor baru. Proses yang sama akan terus dilakukan secara berulang sehingga semua subvektor pada akhirnya hanya tinggal mempunyai satu elemen data. Dalam kondisi demikian, maka semua data dalam vektor K telah menjadi urut naik.

B. *Metode Merge Sort*

Metode *Merge Sort* (penggabungan) melakukan pengurutan dengan cara membagi data menjadi 2 (dua) kumpulan data dan masing-masing kumpulan diurutkan, kemudian setelah terurut dilakukan penggabungan dua kumpulan data tersebut dalam keadaan terurut.

Langkah-langkah pengurutan dengan *Merge Sort* secara garis besar dapat dijelaskan sebagai berikut ini. Sebuah vektor K dengan cacah elemen data sebanyak N pada kondisi tidak terurut. Untuk mengurutkan semua data dalam K, mula-mula setiap elemen dalam vektor K dianggap sebagai sebuah vektor yang masing-masing mempunyai sebuah elemen data. Dengan demikian akan terdapat N vektor dengan cacah elemen masing-masing adalah 1 buah. Selanjutnya, setiap pasang vektor yang berurutan digabungkan menjadi sebuah vektor yang baru. Jika menginginkan hasil secara urut naik, maka pada saat menggabungkan kedua vektor tersebut sekaligus dilakukan perbandingan dan pertukaran posisi antar elemen data sedemikian rupa sehingga data yang lebih kecil akan ditempatkan pada posisi yang mendahului data lain yang lebih besar. Pada akhir langkah pertama ini, akan memiliki vektor baru sebanyak N/2 yang masing-masing dalam kondisi urut naik. Cacah elemen pada masing-masing vektor adalah 2 buah, kecuali jika N bernilai ganjil maka akan terdapat N/2+1 vektor baru dimana salah satu vektor hanya memiliki sebuah elemen data saja.

Pada langkah selanjutnya, setiap pasang vektor yang berurutan dapat dilakukan penggabungan dan sekaligus dapat pertukaran posisi antar data sehingga akan terbentuk vektor-vektor baru. Demikian, proses penggabungan dan pertukaran posisi data secara terus menerus akan dilakukan sehinggga pada akhirnya akan diperoleh sebuah vektor baru yang memuat semua elemen data dalam vektor sumber dalam kondisi urut naik.

C. *Analisa Perbandingan Algoritma*

Pada algoritma *Quick Sort*, jarak dari kedua elemen yang ditukarkan dibuat cukup besar dengan tujuan untuk mempertinggi efektivitasnya. Sedangkan algoritma *Merge Sort* akan selalu membagi dua tiap sub-*array*nya hingga mencapai basis, sehingga kompleksitas dari algoritma *Merge Sort*, berlaku untuk semua kasus (*Worst Case = Best Case = Average Case*). Algoritma *Quick Sort* dan *Merge Sort* memiliki kompleksitas yang sama yaitu *O (n log n)* tetapi pada kasus *worst case* algoritma *Quick Sort* memiliki kompleksitas O (n^2). Metode *Quick Sort* dan metode *Merge Sort* dapat dilakukan dengan menggunakan rekursif atau tanpa rekursif.

Algoritma *Quick Sort* lebih dipilih dibanding algoritma *Merge Sort*, karena algoritma *Merge Sort* melakukan tiga kali lebih banyak *assignment records* dibanding algoritma *Quick Sort* secara rata-ratanya. Walaupun algoritma *Quick Sort* melakukan pembandingan elemen yang lebih banyak dalam kasus rata-ratanya.

Untuk mengetahui analisa perbandingan untuk O (n log n) algoritma pengurutan *Merge Sort* dan *Quick Sort*, dapat kita lihat seperti tabel 3.3 berikut ini :

Tabel 1. Analisa Rangkuman untuk O (n log n) Algoritma Pengurutan Quick Sort dan Merge Sort

Algoritma	**Pembandingan Elemen**
Merge Sort	$W(n) = n \log n$
	$A(n) = n \log n$
Quick Sort	$W(n) = n^2/2$
	$A(n) = 1.38n \log n$
Algoritma	***Assignment Records***
Merge Sort	$T(n) = 2n \log n$
Quick Sort	$A(n) = 0.69n \log n$

VI. PENUTUP

Dari hasil analisis pengurutan data (*Sorting*) menggunakan algoritma *Quick Sort dan Merge Sort* ini, dapat diambil kesimpulan sebagai berikut :

1. Algoritma *Merge Sort* memiliki waktu yang lebih cepat pada jumlah data yang relatif sedikit

(berkisar 1 s/d 10 data), baik pengurutan data pada data *Array* maupun data *Record*.

2. Algoritma *Quick Sort* selalu memiliki jumlah langkah yang sedikit dibandingkan algoritma *Merge Sort* yang memiliki jumlah langkah yang lebih banyak.
3. Dari hasil perbandingan pengurutan data ini, *Quick Sort* dipilih lebih baik dan lebih cepat karena *Assignment Records*nya yang paling sedikit diantara kedua algoritma pengurutan data tersebut, meskipun pembandingan elemen yang dilakukan *Quick Sort* lebih banyak dibandingkan *Merge Sort*.

DAFTAR PUSTAKA

1. Sumantri Slamet I. S, FX. Nursalmi, C. Hendrik, Wahyu, Wibowo, 1989, ***Pengantar Struktur Data***, Penerbit PT. Elex Media Kamputindo, Jakarta.
2. Kusumo Suryo Ario Drs, 2000, ***Microsoft Visual Basic 6***, Penerbit PT. Elex Media Komputindo, Jakarta.
3. Liem, Inggriani, 2007, ***Draft Diktat Kuliah Dasar Pemrograman (Bagian Pemrograman Prosedural)***, Program Studi Teknik Informatika, Institut Teknologi Bandung. hlm. 141-142
4. Munir, Rinaldi, 2006, ***Strategi Algoritmik,*** Jurnal Teknik Informatika Bandung.
5. Jogiyanto MH, Akt, MBA, Ph.D, 2009, ***Bahasa Pascal***, Penerbit Andi Offset, Yogyakarta.