DAFTAR PUSTAKA

Astutik, E. P., & Fitriatien, S. R. (2019). Pengaruh Software Matlab Terhadap Kemampuan Menyelesaikan Masalah Program Linier. *Fibonacci: Jurnal Pendidikan Matematika dan Matematika*, *5*(2), 175. https://doi.org/10.24853/fbc.5.2.175-182

Auliasari, K., Kertaningtyas, M., & Basuki, D. W. L. (2018). Optimalisasi Rute Distribusi Produk Menggunakan Metode Traveling Salesman Problem. *Jurnal Sains, Teknologi dan Industri*, *16*(1), 15. https://doi.org/10.24014/sitekin.v16i1.6109

Basriati, S., Safitri, E., & Anggraini, R. (2021). Optimization of distribution routes in resolving traveling salesman problems using the tabu search algorithm (case study: CV. Bintang anugerah sukses pekanbaru). *Desimal: Jurnal Matematika*, *4*(3), 277–284. https://doi.org/10.24042/djm.v4i3.9811

Davendra, D. (2010). *Traveling Salesman Problem, Theory and Applications. IntechOpen.*

Euchi, J., & Chabchoub, H. (2010). A hybrid tabu search to solve the heterogeneous fixed fleet vehicle routing problem. *Logistics Research*, *2*(1), 3–11. https://doi.org/10.1007/s12159-010-0028-3

Fitri, S. J., Huda, N., & Tasri, E. S. (2016). Optimalisasi Sistem Transportasi Produk Olahan Kacang Pt.Garuda Food Di Kota Solok Dengan Pendekatan Model Quantitative System Business (Studi Kasus Pt. Garuda Food). *Jurnal Fakultas Ekonomi*, *3*. https://ejournal.unsri.ac.id/index.php/mining/article/view/3814.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, *40*(10), 1276–1290. https://doi.org/10.1287/mnsc.40.10.1276

Glover, F. (1990). Tabu Search Part 2. *ORSA Journal on Computing*, *1*(2), 4–32.

Glover, F., & Laguna, M. (1997). *Tabu Search*. Kluwer Acadmeic Publisher.

Gunantara, N. (2018). *Teknik Optimasi* (1 ed.). Udayana University Press.

Gutin, G., & Punnen, A. P. (2006). *The Traveling Salesman Problem and Its Variations*. Springer Science.

Hardian, R. (2022). *Perbandingan Penerapan Algoritma Tabu Search dan Simple Hill Climbing Dalam Mencari Rute Optimal Yang Dilalui Mobil PT. Tiki Jalur Nugraha Ekakurir (JNE) Wilayah Kota Jambi* [Universitas Jambi]. https://repository.unja.ac.id/30188/

Hay's, R. N. (2017). *Implementasi Firefly Algorithm-Tabu Search Untuk Penyelesaian Traveling Salesman Problem*. *2*(1).

Juwita, R. (2012). *Penentuan titik distribusi yang optimal dari perusahaan fast moving consumer goods dengan algoritma tabu search*. Universitas Indonesia.

Kadam, H. B. G., Mulyana, I. J., & Mulyono, J. (2018). Penentuan Rute Terpendek Dengan Metode Tabu Search (Studi Kasus). *Scientific Journal Widya Teknik*, *17*(2).

Kusumadewi, S., & Purnomo, H. (2005). *Penyelesaian Masalah Optimasi dengan Teknik-tekni Heuristik*. Graha Ilmu.

Laporte, G. (1991). The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*, 231–247.

Mutakhiroh, I., & Hidayat, T. (2007). *Pencarian Jalur Terpendek Menggunakan Algoritma Semut*.

Pujiriyanto, A. (2004). *Cepat Mahir Matlab*.

Puspitorini, S. (2008). Penyelesaian Masalah Traveling Salesman Problem dengan Jaringan Saraf Self Organizing. *Media Informatika*, *6*(1), 39–55. https://doi.org/10.20885/informatika.vol6.iss1.art3

Ramadhania, S. E. (2020). *Implentasi Algoritma Genetika dan Tabu Search Untuk Travelling Salesman Problem* [Universitas Islam Indonesia]. https://dspace.uii.ac.id/handle/123456789/31464

Ramadhini, S. (2017). *Perancangan simulator modulasi dan demodulasi M-PSK berbasis MATLAB*. Politeknik Negeri Sriwijaya.

Refianti, R., & A. Benny Mutiara. (2005). *Solusi optimal travelling salesman problem dengan Ant Colony System (ACS)*. https://doi.org/10.13140/RG.2.1.2089.7047

Riswan, Sahari, A., & Lusiyanti, D. (2020). Penentuan Rute Terpendek Pendistribusian Tabung Gas Lpg 3 Kg Pt. Fega Gas Palu Pratama Menggunakan Algoritma Tabu Search. *JURNAL ILMIAH MATEMATIKA DAN TERAPAN*, *16*(2), 221–229. https://doi.org/10.22487/2540766X.2019.v16.i2.15004

Saptono, F., & Hidayat, T. (2007). *Perancangan Algoritma Genetika Untuk Menentukan Jalur Terpendek*.

Simarmata, J. E. (2020). Penerapan Algoritma Branch And Bound Pada Persoalan Pedagang Keliling (Travelling Salesman Problem). *RANGE: Jurnal Pendidikan Matematika*, *1*(2), 111–121. https://doi.org/10.32938/jpm.v1i2.366

Sugiyono. (2018). *Metode Penelitian Kuantitatif, Kualitatif, R&D*. Alfabeta.

Sukardi, D. (2009). *Manajemen Distribusi "Old Distribution Channel And Postmo Distribution Channel Approach."* Graha Ilmu.

Sutoyo, I. (2018). *Penerapan Algoritma Nearest Neighbour untuk Menyelesaikan Travelling Salesman Problem*. *1*.

Suyanto. (2010). *Algoritma Optimasi Deterministik atau Probabilistik*. Graha Ilmu.

Taha, H. A. (1996). *Riset Operasi* (1 ed., Vol. 2). Binarupa Aksara.

Wiyanti, D. T. (2013). Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem. *Jurnal Transformatika*, *11*(1), 1. https://doi.org/10.26623/transformatika.v11i1.76

Yulianto, E., & Setiawan, A. (2018). Optimasi Rute Sales Coverage Menggunakan Algoritma Cheapest Insertion Heuristic Dan Layanan Google Maps Api. *Internal (Information System Journal)*, *1*(1), 39–54. https://doi.org/DOI:10.32627/internal.v1i1.326

LAMPIRAN

Matlab Code

```
function tabusearch(d,Maxit);
d=xlsread('mapspjpcsosenin.xlsx','B2:W23');
Maxit=100;

clc

n=length(d);
ActionList= permaction(n) % daftar aksi
nAction= numel(ActionList); % jumlah aksi
TL= round(0.5*nAction); % Tabu Length

%% Initialization
% Create Empty Individual Structure
empty_individual.Rute= [];
empty_individual.Biaya= [];

% Bangkitkan solusi awal
sol= empty_individual;
sol.Rute= randperm(n);


    sol.Rute(sol.Rute==1) = []; % kembali ke rute awal
    sol.Rute = [1,sol.Rute,1];

sol.Biaya= jartsp([sol.Rute sol.Rute(1)],d);
 % jarak total
 BestSol= sol;

% Array untuk menyimpan biaya terkecil
Bestbiaya= zeros(Maxit,1);
% vektor untuk menyimpan jumlah tabu
TC= zeros(nAction,1);


%% Tabu Search Main Loop
for it=1:Maxit;
    ts=tic;
    rand(10);
    bestnewsol.Biaya=inf;
    % Terapkan aksi
    for i=1:nAction
        if TC(i)==0
            newsol.Rute=DoAction(sol.Rute,ActionList{i}); % ubah
solusi
            newsol.Biaya= jartsp([newsol.Rute newsol.Rute(1)],d);
% hitung jarak solusi baru
            newsol.ActionIndex=i;
            if newsol.Biaya<= bestnewsol.Biaya
```

```
                bestnewsol= newsol;
            end
        end
    end

    % Update solusi sekarang
    sol= bestnewsol;

    % Update tabu list
    for i=1:nAction
        if i== bestnewsol.ActionIndex
            TC(i)= TL; % Add to tabu list
            TC(i)= max(TC(i)-1,0); % Reduce tabu counter
        end
    end
     disp(' ')
     texty = sprintf('% .0f',TC);
     disp(texty)
     disp(' ')

    % Update best solution ever found
    if sol.Biaya<BestSol.Biaya
        BestSol=sol;
        Tit=it;
        Time=toc(ts);
        Solusi=BestSol.Rute;
        Action=BestSol.ActionIndex;
    end

    % Save best biaya ever found
    BestBiaya(it)= BestSol.Biaya;


    % Show Iteration Information
    textt = sprintf('% .0f',Solusi);
    disp(['Iteration ' num2str(it) ': Best Cost = '
num2str(BestBiaya(it)) ' : Best Sol =' textt ' : Total Moves = '
num2str(Action) ' : Time = ' num2str(Time)]);

    % If Global Minimun is Reached
    if BestBiaya(it)==0
        break;
    end
end
disp(' ');
disp(['Best Cost terdapat pada iterasi ke ' num2str(Tit) ' sebesar
= ' num2str(BestBiaya(it)) ' dengan langkah sebanyak '
num2str(Action) ' dan lama waktu ' num2str(Time) ' detik']);
disp(' ');
disp(['Beserta Best Solnya yaitu = ' textt]);
BestBiaya= BestBiaya(1:it);

figure;
plot(BestBiaya,'LineWidth',2);
xlabel('Iteration');
```

```
ylabel('Best Cost');
grid on;

 function ActionList= permaction(n)
        nSwap= n*(n-1)/2;
        nFlip= n*(n-1)/2;
        nSlide= n^2;
        nActionList= nSwap+nFlip+nSlide;
       % ActionList= cell(nAction,1);
        c=0;

       % Add swap
       for i=2:n
             for j=i+1:n


                c=c+1;
                ActionList{c}= [1 i j];
            end
        end

       % Add Flip


         for i=2:n
            for j=i+1:n
                if abs(i-j)>2
                    c=c+1;
                    ActionList{c}= [2 i j];
                end
            end
        end


        for i=2:n
            for j=2:n

                if abs(i-j)>1
                    c=c+1;
                    ActionList{c}=[3 i j];
                end
            end
        end
        ActionList= ActionList(1:c);


        function q= DoAction(p,a)
        switch a(1)
            case 1
           %     swap
                q=DoSwap(p,a(2),a(3));
            case 2
             %   Flip
                q= DoFlip(p,a(2),a(3));
```

```
            case 3
             %   slide
                q= DoSlide(p,a(2),a(3));
        end

            function q= DoSwap(p,i1,i2)
                q=p;
                q([i1 i2])= p([i2,i1]);

            function q=DoSlide(p,i1,i2)
                if i1<i2
                    q=p([1:i1-1 i1+1:i2 i1 i2+1:end]);
                else
                    q=p([1:i2 i1 i2+1:i1-1 i1+1:end]);
                end

            function q=DoFlip(p,i1,i2)
                q=p;
                if i1<i2
                    q(i1:i2)=p(i2:-1:i1);
                else
                    q(i1:-1:i2)=p(i2:i1);
                end


function jarak=jartsp(x1,dx)
[r,c]=size(x1);
k=c-1; %jumlah kota dalam rute tsp
s=0; %jarak awal di kota pertama
for j=1:k
    s=s+dx(x1(j),x1(j+1)); %pengakumulasian jarak rute tsp
end
jarak=s;
```