```
% rewrited by wang.david.wei 2020.3.5
% 2020.3.6 compress with start position
function mtspf_ga(varargin)
    xy          =   100*rand(61,2); % 60 sensor nodes position and start/end position
    xy(1,:)     =   [0 0];          %start/end position(0,0)
    dmat        =   [];             %distance matrix for all sensors
    nSalesmen   =   4;              %mutiple salesmen
    minTour     =   2;              %min tour constraint
    popSize     =   80;             %particle size or population
    numIter     =   5e3;            %the number of iterate
    showProg    =   true;           %show the figure

    if isempty(dmat)                %distance matrix calculation
        nPoints = size(xy,1);       %distance matrix size
        a = meshgrid(1:nPoints);    %meshgrid
        dmat = reshape(sqrt(sum((xy(a,:)-xy(a',:)).^2,2)),nPoints,nPoints);
    end

    N=nPoints;  %senors and start/end position sum
    n   =N-1;   %sensor number
    nSalesmen   = max(1,min(n,round(real(nSalesmen(1)))));
    minTour     = max(1,min(floor(n/nSalesmen),round(real(minTour(1)))));
    popSize     = max(8,8*ceil(popSize(1)/8));
    numIter     = max(1,round(real(numIter(1))));
    showProg    = logical(showProg(1));
    % Initializations for Route Break Point Selection
    nBreaks = nSalesmen-1;
    dof = n - minTour*nSalesmen;          % degrees of freedom
    addto = ones(1,dof+1);
    for k = 2:nBreaks
        addto = cumsum(addto);
    end
    cumProb = cumsum(addto)/sum(addto);
    % Initialize the Populations
    popRoute = zeros(popSize,n);         % population of routes
    popBreak = zeros(popSize,nBreaks);   % population of breaks
    popRoute(1,:) = (1:n) + 1;
    popBreak(1,:) = rand_breaks();
    for k = 2:popSize
        popRoute(k,:) = randperm(n) + 1;
        popBreak(k,:) = rand_breaks();
    end
    pclr = ~get(0,'DefaultAxesColor');  % Select the Colors for the Plotted Routes
    clr = [1 0 0; 0 0 1; 0.67 0 1; 0 1 0; 1 0.5 0];
    globalMin = Inf; % Run the GA
    totalDist = zeros(1,popSize);
    distHistory = zeros(1,numIter);
    tmpPopRoute = zeros(8,n);
    tmpPopBreak = zeros(8,nBreaks);
    newPopRoute = zeros(popSize,n);
    newPopBreak = zeros(popSize,nBreaks);
    if showProg
        figure('Name','MTSP for WRSN| Current Best Solution','Numbertitle','off');
        hAx = gca;
    end
    for iter = 1:numIter
        for p = 1:popSize    % Evaluate Members of the Population
            d = 0;
            pRoute = popRoute(p,:);
            pBreak = popBreak(p,:);
            rng = [[1 pBreak+1];[pBreak n]]';
            for s = 1:nSalesmen
                d = d + dmat(1,pRoute(rng(s,1))); % Add Start Distance
                for k = rng(s,1):rng(s,2)-1
                    d = d + dmat(pRoute(k),pRoute(k+1));
                end
                d = d + dmat(pRoute(rng(s,2)),1); % Add End Distance
            end
            totalDist(p) = d;
        end
        [minDist,index] = min(totalDist);  % Find the Best Route in the Population
        distHistory(iter) = minDist;
        if minDist < globalMin
```

```
74                globalMin = minDist;
75                optRoute = popRoute(index,:);
76                optBreak = popBreak(index,:);
77                rng = [[1 optBreak+1];[optBreak n]]';
78                if showProg % Plot the Best Route
79                    for s = 1:nSalesmen
80                        rte = [1 optRoute(rng(s,1):rng(s,2)) 1];
81                        plot(hAx,xy(rte,1),xy(rte,2),'.-','Color',clr(s,:));
82                        hold(hAx,'on');
83                    end
84                    plot(hAx,xy(1,1),xy(1,2),'o','Color',pclr);
85                    title(hAx,sprintf('Total Distance = %1.4f, Iteration =
                      %d',minDist,iter));
86                    hold(hAx,'off');
87                    drawnow;
88                end
89            end
90            randomOrder = randperm(popSize); % Genetic Algorithm Operators
91            for p = 8:8:popSize
92                rtes = popRoute(randomOrder(p-7:p),:);
93                brks = popBreak(randomOrder(p-7:p),:);
94                dists = totalDist(randomOrder(p-7:p));
95                [ignore,idx] = min(dists); %#ok
96                bestOf8Route = rtes(idx,:);
97                bestOf8Break = brks(idx,:);
98                routeInsertionPoints = sort(ceil(n*rand(1,2)));
99                I = routeInsertionPoints(1);
100               J = routeInsertionPoints(2);
101               for k = 1:8 % Generate New Solutions
102                   tmpPopRoute(k,:) = bestOf8Route;
103                   tmpPopBreak(k,:) = bestOf8Break;
104                   switch k
105                       case 2 % Flip
106                           tmpPopRoute(k,I:J) = tmpPopRoute(k,J:-1:I);
107                       case 3 % Swap
108                           tmpPopRoute(k,[I J]) = tmpPopRoute(k,[J I]);
109                       case 4 % Slide
110                           tmpPopRoute(k,I:J) = tmpPopRoute(k,[I+1:J I]);
111                       case 5 % Modify Breaks
112                           tmpPopBreak(k,:) = rand_breaks();
113                       case 6 % Flip, Modify Breaks
114                           tmpPopRoute(k,I:J) = tmpPopRoute(k,J:-1:I);
115                           tmpPopBreak(k,:) = rand_breaks();
116                       case 7 % Swap, Modify Breaks
117                           tmpPopRoute(k,[I J]) = tmpPopRoute(k,[J I]);
118                           tmpPopBreak(k,:) = rand_breaks();
119                       case 8 % Slide, Modify Breaks
120                           tmpPopRoute(k,I:J) = tmpPopRoute(k,[I+1:J I]);
121                           tmpPopBreak(k,:) = rand_breaks();
122                       otherwise % Do Nothing
123                   end
124               end
125               newPopRoute(p-7:p,:) = tmpPopRoute;
126               newPopBreak(p-7:p,:) = tmpPopBreak;
127           end
128           popRoute = newPopRoute;
129           popBreak = newPopBreak;
130       end
131       function breaks = rand_breaks() % Force Breaks to be at Least the Minimum Tour
          Length
132           nAdjust = find(rand < cumProb,1)-1;
133           spaces = ceil(nBreaks*rand(1,nAdjust));
134           adjust = zeros(1,nBreaks);
135           for kk = 1:nBreaks
136               adjust(kk) = sum(spaces == kk);
137           end
138           breaks = minTour*(1:nBreaks) + cumsum(adjust);
139       end
140   end
```