```
%
% rewrited by wang.david.wei 2020.3.5
function varargout = mtspf_ga(varargin)
    xy          = 10*rand(60,2);
    dmat        =[];
    nSalesmen   = 4;
    minTour     =2;
    popSize     =80;
    numIter     =5e3;
    showProg    =true;
    if isempty(dmat)
        nPoints = size(xy,1);
        a = meshgrid(1:nPoints);
        dmat = reshape(sqrt(sum((xy(a,:)-xy(a',:)).^2,2)),nPoints,nPoints);
    end
    [N,dims]    = size(xy);
    [nr,nc]     = size(dmat);
    n   =N-1;
    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
        f11=figure('Name','MTSPF_GA | 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
            globalMin = minDist;
            optRoute = popRoute(index,:);
```

```
            optBreak = popBreak(index,:);
            rng = [[1 optBreak+1];[optBreak n]]';
            if showProg % Plot the Best Route
                for s = 1:nSalesmen
                    rte = [1 optRoute(rng(s,1):rng(s,2)) 1];
                    plot(hAx,xy(rte,1),xy(rte,2),'.-','Color',clr(s,:));
                    hold(hAx,'on');
                end
                plot(hAx,xy(1,1),xy(1,2),'o','Color',pclr);
                title(hAx,sprintf('Total Distance = %1.4f, Iteration =
                %d',minDist,iter));
                hold(hAx,'off');
                drawnow;
            end
        end
        randomOrder = randperm(popSize); % Genetic Algorithm Operators
        for p = 8:8:popSize
            rtes = popRoute(randomOrder(p-7:p),:);
            brks = popBreak(randomOrder(p-7:p),:);
            dists = totalDist(randomOrder(p-7:p));
            [ignore,idx] = min(dists); %#ok
            bestOf8Route = rtes(idx,:);
            bestOf8Break = brks(idx,:);
            routeInsertionPoints = sort(ceil(n*rand(1,2)));
            I = routeInsertionPoints(1);
            J = routeInsertionPoints(2);
            for k = 1:8 % Generate New Solutions
                tmpPopRoute(k,:) = bestOf8Route;
                tmpPopBreak(k,:) = bestOf8Break;
                switch k
                    case 2 % Flip
                        tmpPopRoute(k,I:J) = tmpPopRoute(k,J:-1:I);
                    case 3 % Swap
                        tmpPopRoute(k,[I J]) = tmpPopRoute(k,[J I]);
                    case 4 % Slide
                        tmpPopRoute(k,I:J) = tmpPopRoute(k,[I+1:J I]);
                    case 5 % Modify Breaks
                        tmpPopBreak(k,:) = rand_breaks();
                    case 6 % Flip, Modify Breaks
                        tmpPopRoute(k,I:J) = tmpPopRoute(k,J:-1:I);
                        tmpPopBreak(k,:) = rand_breaks();
                    case 7 % Swap, Modify Breaks
                        tmpPopRoute(k,[I J]) = tmpPopRoute(k,[J I]);
                        tmpPopBreak(k,:) = rand_breaks();
                    case 8 % Slide, Modify Breaks
                        tmpPopRoute(k,I:J) = tmpPopRoute(k,[I+1:J I]);
                        tmpPopBreak(k,:) = rand_breaks();
                    otherwise % Do Nothing
                end
            end
            newPopRoute(p-7:p,:) = tmpPopRoute;
            newPopBreak(p-7:p,:) = tmpPopBreak;
        end
        popRoute = newPopRoute;
        popBreak = newPopBreak;
    end
  function breaks = rand_breaks()
    if minTour == 1 % No Constraints on Breaks
        tmpBreaks = randperm(n-1);
        breaks = sort(tmpBreaks(1:nBreaks));
    else % Force Breaks to be at Least the Minimum Tour Length
        nAdjust = find(rand < cumProb,1)-1;
        spaces = ceil(nBreaks*rand(1,nAdjust));
        adjust = zeros(1,nBreaks);
        for kk = 1:nBreaks
            adjust(kk) = sum(spaces == kk);
        end
        breaks = minTour*(1:nBreaks) + cumsum(adjust);
    end
  end
end
```