

Solve traveling salesman problem using EMF-CE algorithm

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

24-10-2020 / 28-10-2020

CITATION

Luo, Meng; Gu, Shiliang (2020): Solve traveling salesman problem using EMF-CE algorithm. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.13139042.v2>

DOI

[10.36227/techrxiv.13139042.v2](https://doi.org/10.36227/techrxiv.13139042.v2)

C.EXCHANGE, MOVE, AND FLIP

Algorithm1 Move $1 \rightarrow n$

```

1:while reap<=3 && nd<10 % starting set the reap=0 and nd=0
2:   for i=1:n-2
3:     for j=i+2:n-1
4:       calculate the  $Td_1, Td_2$  and also  $Td=Td_2-Td_1$ 
5:       if  $Td < C_{new}$  %setting the C(critical value) and nd  $\in [0,10)$ 
6:         then move all of the satisfied  $v_i$  or  $v_j$ 
7:           if made shorter tour than before % greater improvement
8:             then Calculate the shortest paths through and get the new TSP
9:           end if
10:        end if
11:      end for
12:    end for
13:    set reap=reap+1,nd=nd+1
14:end while

```

Move $1 \leftarrow n$

```

1:while reap<=3 && nd<10 %also starting set the reap=0 and nd=0
2:   for i=n+1:-1:3
3:     for j=i-2:-1:2
4:       also calculate the  $Td_1, Td_2$  and also  $Td=Td_2-Td_1$ 
5:       if  $Td < C_{new}$  %setting the C and nd  $\in [0,10)$ 
6:         then move the nodes of satisfied  $v_i, v_j$ 
7:           if made shorter tour than before % greater improvement
8:             then reap the new TSP %TSP is the length of optimal
9:           end if
10:        end if
11:      end for
12:    end for
13:    set reap=reap+1,nd=nd+1
14:end while

```

Algorithm2 exchange $1 \leftarrow n$

```

1:for r=1:3
2:   n1=max(randi(n+1,3,10)) % the randi was uniformly distributed pseudorandom integers
3:   for i=n1:-1:3
4:     if i<n+1
5:       calculate the  $Td_1, Td_2$  and also  $Td=Td_2-Td_1$ 
6:       if  $Td < 0$ 
7:         then move all of the satisfied  $v_i$  or  $v_j$ 
8:           if made shorter tour than before % greater improvement
9:             then reap the new TSP %the length of optimal than before
10:            end if
11:          end if
12:        end if
13:      end for
14:    end for
15:    for j=i-3:-1:2
16:      calculate the  $Td_1, Td_2$  and also  $Td=Td_2-Td_1$ 
17:      if  $Td < C_{new}$ 
18:        then move the nodes of satisfied  $v_i, v_j$ 
19:          if made shorter tour than before % greater improvement

```

```

20:           then reap the new TSP      %TSP was the length of optimal
21:       end if
22:   end if
23: end for

```

Algorithm3 Move+Crt $I \rightarrow n$

```

1:For r=1:3
2: n1=min(randi(fix(n/3),1,10)) % the randi was uniformly distributed pseudorandom integers
3: for i=n1:n-2
4:   for j=i+2:n
5:     calculate the  $Td_1, Td_2$  and also  $Td=Td_2-Td_1$ 
6:     if  $Td < C_{new}$ 
7:       then move all of the satisfied  $v_i$  or  $v_j$ 
8:         if made shorter tour than before % greater improvement
9:           then Calculate the shortest paths through and reap the new TSP(the length optimal)
10:          end if
11:        end if
12:      end for
13:    end for
14:end for

```

Algorithm4 Move+Crt $I \rightarrow n$

```

1:For r=1:3
2: n1=min(randi(fix(n/3),1,10)) % the randi was uniformly distributed pseudorandom integers
3: for i=n1:n-3
4:   for j=i+2:n-2
5:     calculate the  $Td_1, Td_2, Td_3$ 
6:     if  $Td_3 < Td_2, Td_3=Td_2, id=3$  else  $id=2$  end if and also  $Td=Td_2-Td_1$ 
7:     if  $Td < C_{new}$ ,then new shorter tour and the lengthof optimal, set the new critical value
7:       if  $id=2$ , then used the move  $Td_2$  else used the move  $Td_3$  end if
8:         if made shorter tour than before % greater improvement
9:           then Calculate the shortest paths through and reap the new TSP(the length optimal)
10:          end if
11:        end if
12:      end for
13:    end for
14:end for

```

Algorithm5 Move+Crt $\leftarrow n$

```

1:For r=1:3
2: n1=max(randi(n+1,1,10)) % the randi was uniformly distributed pseudorandom integers
3: for i=n1:-1:6
4:   for j=i-2:-1:4
5:     calculate the  $Td_1, Td_2, Td_3$ 
6:     if  $Td_3 < Td_2, Td_3=Td_2, id=3$  else  $id=2$  end if then also calculates the  $Td=Td_2-Td_1$ 
7:     if  $Td < C_{new}$ ,then new shorter tour and the lengthof optimal, set the new critical value
7:       if  $id=2$ , then used the move  $Td_2$  else used the move  $Td_3$  end if
8:         if made shorter tour than before % greater improvement
9:           then Calculate the shortest paths through and reap the new TSP(the length optimal)
10:          end if

```

```
11:      end if
12:    end for
13:  end for
14:end for
```

Algorithm6 Move 1→n

```
1:while reap<=2 % starting set the reap=0
2: by using the Algorithm1 Move 1→n ①
3:end while
```

Algorithm7 Move 1→n

```
If mod(v1,2)==1 %setting the condition of satisfied
1:For r=1:3
2: n1=min(randi(fix(n/3),1,10)) % the randi was uniformly distributed pseudorandom integers
3: for i=n1:n-5
4:   for j=i+2:n-3
5:     calculate the Td1, Td2, Td3
6:     if Td3<Td2, Td3=Td2, id=3 else id=2 end if then also calculates the Td=Td2-Td1
7:     if Td<Cnew ,then new shorter tour and the lengthof optimal,set the new critical value
7:     if id=2, then used the move Td2 else used the move Td3 end if
8:       if made shorter tour than before % greater improvement
9:         then Calculate the shortest paths through and reap the new TSP(the length optimal)
10:        end if
11:      end if
12:    end for
13:  end for
14:end for
```

Move+Crt 1→n

```
1:For r=1:3
2: n1=min(randi(fix(n/3),1,10)) % the randi was uniformly distributed pseudorandom integers
3: for i=n1:n-6
4:   for j=i+2:n-4
5:     calculate the Td1, Td2, Td3
6:     if Td3<Td2, Td3=Td2, id=3 else id=2 end if then also calculates the Td=Td2-Td1
7:     if Td<Cnew ,then new shorter tour and the length of optimal,set the new critical value
7:     if id=2, then used the move Td2 else used the move Td3 end if
8:       if made shorter tour than before % greater improvement
9:         then Calculate the shortest paths through and reap the new TSP(the length optimal)
10:        end if
11:      end if
12:    end for
13:  end for
14:end for
end if
```

Algorithm8 Move 1————n

```

1:while reap<=2 % starting set the reap=0
2: by using the Algorithm1 Move 1————n ①
3:end while

```

Algorithm9 Move 1————n

```

1:while reap<=2 % starting set the reap=0
2: by using the Algorithm1 Move 1————n ②
3:end while

```

Algorithm10 Move +C 1————n

```

1:For r=1:3
2:   for i=1:n-4
3:     for j=i+2:n-3
4:       calculate the  $Td_1, Td_2, Td_3$ 
5:       if  $Td_3 < Td_2$ ,  $Td_3 = Td_2$ , id=3 else id=2 end if then also calculates the  $Td = Td_2 - Td_1$ 
6:       if  $Td < C_{new}$  ,then new shorter tour and the length of optimal, set the new critical value
7:       if id=2, then used the move  $Td_2$  else used the move  $Td_3$  end if
8:         if made shorter tour than before % greater improvement
9:           then Calculate the shortest paths through and reap the new TSP(the length optimal)
10:          end if
11:        end if
12:      end for
13:    end for
14:end for

```

Algorithm11 Move 1————n

```

1:while reap<=2 && nd<=5 % starting set the reap=0 && nd=0
2: By using the Algorithm1 Move 1————n ① and change the critical value
12:end while

```

For this search, which we also using as same as algorithm 1 move. But the critical value should be the following:

$$Td < C_{new} \quad (C_{new} = 0.25 \times C_{old} / \sqrt{0.25 + nd}, nd \in [0, 5])$$

The size of the indicated variates or array appears to be changing with each loop iteration.

Algorithm12 Move +C $I \rightarrow n$

```
1:For r=1:3
2:   for i=n+1:-1:5
3:     for j=i-2:-1:3
4:       calculate the  $Td_1, Td_2, Td_3$ 
5:       if  $Td_3 < Td_2$ ,  $Td_3 = Td_2$ , id=3 else id=2 end if then also calculates the  $Td = Td_2 - Td_1$ 
6:       if  $Td < C_{new}$  ,then new shorter tour and the length of optimal, set the new critical value
7:       if id=2, then used the move  $Td_2$  else used the move  $Td_3$  end if
8:       if made shorter tour than before % greater improvement
9:         then Calculate the shortest paths through and repeat the new TSP(the length optimal)
10:        end if
11:      end if
12:    end for
13:  end for
14:end for
```

Algorithm13 Move $I \leftarrow n$

```
1:For r=1:3
2:   by using the Algorithm1 Move  $I \leftarrow n$  ②
3:end for
```

Algorithm14 exchange+C $\cancel{I} \rightarrow n$

```
1:For r=1:3
2:   for i=1:n-2
3:     if i>=2
4:       then calculate the  $Td_1, Td_2, Td = Td_2 - Td_1$ 
5:       if  $Td < 0$ , then used the move  $Td_2$ 
6:       if made shorter tour than before % greater improvement
7:         then Calculate the new shortest paths TSP(the length optimal)
8:       end if
10:      end if
11:      end if
12:    for j=i+3:n
13:      then calculate the  $Td_1, Td_2, Td = Td_2 - Td_1$ 
14:      if  $Td < C_{new}$  , then used the move  $Td_2$ 
15:        if made shorter tour than before % greater improvement
16:          then Calculate the new shortest paths TSP(the length optimal)
17:        end if
18:      end if
19:    end for
20:  end for
21:end for
```

Algorithm15 Move +C $I \leftarrow n$

```
1:For r=1:3
2:   for i=n+1:-1:7
3:     for j=i-2:-1:5
4:       then calculate the  $Td_1, Td_2, Td_3$ 
5:       if  $Td_3 < Td_2$ ,  $Td_3 = Td_2$ , id=3 else id=2 end if then also calculates the  $Td = Td_2 - Td_1$ 
6:       if  $Td < C_{new}$  ,then new shorter tour and the lengthof optimal
7:       if id=2, then used the move  $Td_2$  else used the move  $Td_3$  end if
8:       if made shorter tour than before % greater improvement
9:       then Calculate the shortest paths TSP(the length optimal)
10:      end if
11:      end if
12:    end for
13:  end for
14:end for
15:for r=1:3
16:  for i=n+1:-1:8
17:    for j=i-2:-1:6
18:      then calculate the  $Td_1, Td_2, Td_3$ 
19:      if  $Td_3 < Td_2$ ,  $Td_3 = Td_2$ , id=3 else id=2 end if then also calculates the  $Td = Td_2 - Td_1$ 
20:      if  $Td < C_{new}$  ,then new shorter tour and the lengthof optimal
21:      if id=2, then used the move  $Td_2$  else used the move  $Td_3$  end if
22:      if made shorter tour than before % greater improvement
23:      then Calculate the shortest paths TSP(the length optimal)
24:      end if
25:      end if
26:    end for
27:  end for
28:end for
```

Algorithm16 Move $I \rightarrow n$

```
1:while reap<=2 &&nd<=5 %starting set the reap=0 and nd=0
2: by using the Algorithm1 Move  $I \rightarrow n$  ①
3:end while
```

Algorithm17 Move $I \leftarrow n$

```
1:while reap<=2 &&nd<=5 %starting set the reap=0 and nd=0
2: by using the Algorithm1 Move  $I \leftarrow n$  ②
3:end while
```

Algorithm18 flip $I \rightarrow n$

```
if mod(v1,2)==0
1: while reap<=1      %starting set the reap=0
2:   for i=1:n-2,j=2  %set  $I \rightarrow n$ 
3:     while j<=n/1.5+sqrt(mod(v1,144)) && i+j<n  %set the possible of loop times
4:       then calculate the  $Td_1, Td_2$  and calculates the  $Td=Td_2-Td_1$ 
5:       if  $Td < C_{new}$ 
6:         then flipud and fliplr these nodes
7:         if made shorter tour than before    % greater improvement
8:           then anew calculate the shortest paths TSP(the length optimal)
9:         end if
10:      end if
11:      set j=j+1
12:    end while
13:  end for
14:  set reap=reap+1
15:end while
else
16: while reap<=1      %starting set the reap=0
17:   for i=n+1:-1:3,j=2  %set  $n \rightarrow I$ 
18:     while j<=n/1.5+sqrt(mod(v1,169)) && i-j>2  %set the possible of loop times
19:       then calculate the  $Td_1, Td_2$  and calculates the  $Td=Td_2-Td_1$ 
20:       if  $Td < C_{new}$ 
21:         then flipud and fliplr these nodes
22:         if made shorter tour than before    % greater improvement
23:           then anew calculate the shortest paths TSP (the length optimal)
24:         end if
25:       end if
26:       set j=j+1
27:     end while
28:   end for
29:   set reap=reap+1
30:end while
end
```
