Decomposition for Large Scale Capacitated Arc Routing Problem

Yi Mei [1] *yi.mei@rmit.edu.au*
Xiaodong Li [1] *xiaodong.li@rmit.edu.au*
Xin Yao [2] *x.yao@cs.bham.ac.uk*

RMIT University, Australia

University of Birmingham, UK

June 13, 2014

Outline

Outline

Winter Gritting in UK

- 3000 gritting routes (2006)
- 120,000 km or 30% of the entire road network (2006)
- Millions of pounds each year (2006)

FIGURE: Black ice hazard

Example of South Gloucester, UK

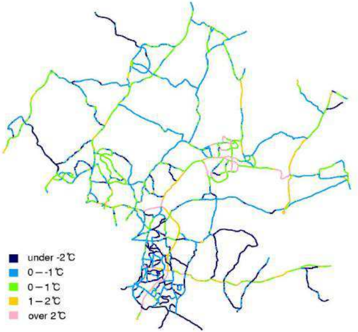


FIGURE: Temperature distribution

Example of South Gloucester, UK

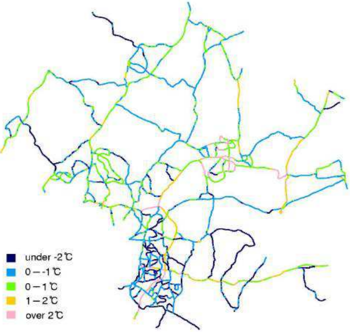


FIGURE: Temperature distribution

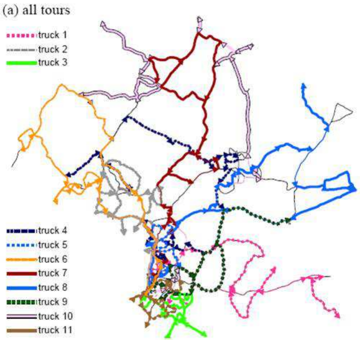


FIGURE: A routing plan

Problem Description

Design a routing plan for a fleet of vehicles to serve the predefined edges (tasks) of the given graph with the minimal total cost so that:

- Each vehicle must start and end at the depot;
- Each task is served exactly once by a vehicle (but can be traversed any times);
- The total demand of the tasks served by each vehicle cannot exceed its capacity.

Problem Description

Design a routing plan for a fleet of vehicles to serve the predefined edges (tasks) of the given graph with the minimal total cost so that:

- Each vehicle must start and end at the depot;
- Each task is served exactly once by a vehicle (but can be traversed any times);
- The total demand of the tasks served by each vehicle cannot exceed its capacity.

NP-hard!

Other Real-world Applications

- Snow removal
- Waste collection
- Post delivery
- ...

Curse of Dimensionality in CARP

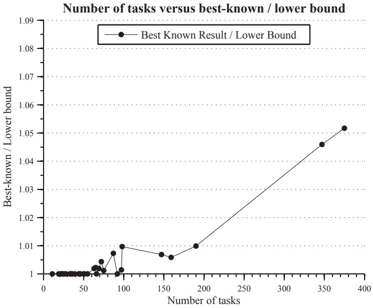


Figure: Size versus best-known / lower bound

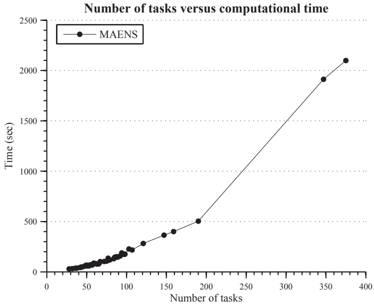


Figure: Size versus computational time of MAENS

Curse of Dimensionality in CARP

- Commonly used benchmark instances (*gdb*, *val*, *egl*, Beullens' C, D, E and F sets) have no larger than 200 tasks;
- Performance of algorithms significantly deteriorate when problem size becomes larger than 300, in terms of both quality of solution and computational time;

Curse of Dimensionality in CARP

- Commonly used benchmark instances (*gdb*, *val*, *egl*, Beullens' C, D, E and F sets) have no larger than 200 tasks;
- Performance of algorithms significantly deteriorate when problem size becomes larger than 300, in terms of both quality of solution and computational time;

Large Scale CARP (LSCARP): the CARPs with more than 300 tasks

Outline

Divide-and-Conquer Strategy

$$
\begin{aligned}
& \mathcal{P}(x_1, x_2, x_3, x_4): \\
\min\ & 3x_1 + 5x_2 + x_3 + 8x_4 \\
s.t.\ & x_1 + 2x_2 \leq 5 \\
& 2x_1 + 3x_2 \leq 6 \\
& x_3 + 3x_4 \leq 10 \\
& 2x_3 + 5x_4 \leq 15
\end{aligned}
$$

DIVIDE-AND-CONQUER STRATEGY

$$
\begin{aligned}
&\mathcal{P}(x_1, x_2, x_3, x_4):\\
\min\ & 3x_1 + 5x_2 + x_3 + 8x_4\\
s.t.\ & x_1 + 2x_2 \leq 5\\
& 2x_1 + 3x_2 \leq 6\\
& x_3 + 3x_4 \leq 10\\
& 2x_3 + 5x_4 \leq 15
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{P}_1(x_1, x_2):\\
\min\ & 3x_1 + 5x_2\\
s.t.\ & x_1 + 2x_2 \leq 5\\
& 2x_1 + 3x_2 \leq 6
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{P}_2(x_3, x_4):\\
\min\ & x_3 + 8x_4\\
s.t.\ & x_3 + 3x_4 \leq 10\\
& 2x_3 + 5x_4 \leq 15
\end{aligned}
$$

Divide-and-conquer in Evolutionary Computation: Cooperative Co-evolution

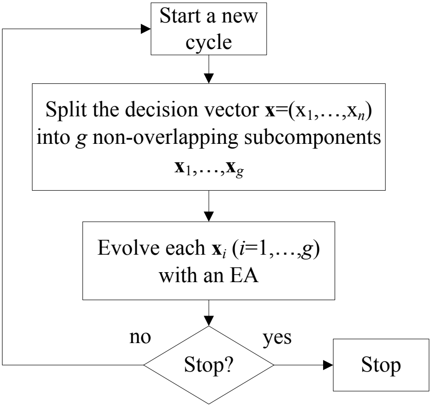


Figure: Generic Cooperative Co-evolution framework

Cooperative Co-evolution (CC) for LSCARP

Generic CC

- Decision variables: $x \in \mathbf{x}$
- Split the decision vector $\mathbf{x}$ into sub-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_g$

Cooperative Co-evolution (CC) for LSCARP

Generic CC

- Decision variables: $x \in \mathbf{x}$
- Split the decision vector $\mathbf{x}$ into sub-vectors $\mathbf{x}_1, \ldots, \mathbf{x}_g$

CC for LSCARP

- Decision variables: Tasks $z \in Z$
- Partition the task set Z into non-overlapping subcomponents $Z_1, \ldots, Z_g$ $(\cup_{i=1}^{g} Z_i = Z)$

Overall problem

Cluster and order all the tasks $z \in Z$ to minimize the total cost so that:

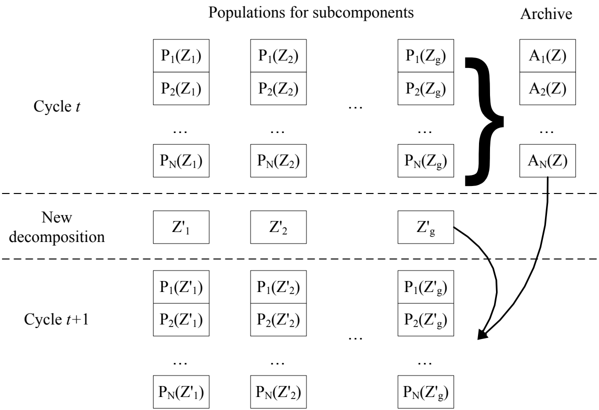
- Each vehicle must starts and ends at the depot;
- Each $z \in Z$ is served exactly once;
- Total demand of each vehicle cannot exceed its capacity.

Decomposed sub-problems

Cluster and order the tasks $z \in Z_i (i = 1, \ldots, g)$ to minimize the total cost so that:

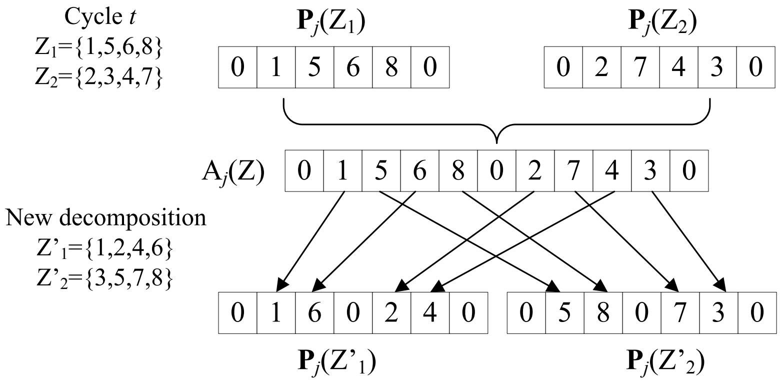
- Each vehicle must starts and ends at the depot;
- Each $z \in Z_i$ is served exactly once;
- Total demand of each vehicle cannot exceed its capacity.

Flowchart of CC for LSCARP



A Simple Example

- Eight tasks: $1, \ldots, 8$ (0 is the depot)
- Two subcomponents: Z_1 and Z_2



Outline

OBTAINING $Z_1, \dots, Z_g$

OBJECTIVE OF DECOMPOSITION

Minimize the total cost of the optimal solutions of the subcomponents:

$$\min_{(Z_i,\dots,Z_g)} \sum_{i=1}^{g} tc(\mathbf{s}^*(Z_i)) \tag{1}$$

- $tc(\cdot)$: total cost function
- $\mathbf{s}^*(Z_i)$: optimal solution for task set Z_i

Obtaining $Z_1, \ldots, Z_g$

Objective of decomposition

Minimize the total cost of the optimal solutions of the subcomponents:

$$\min_{(Z_i,\ldots,Z_g)} \sum_{i=1}^{g} tc(\mathbf{s}^*(Z_i)) \tag{1}$$

- $tc(\cdot)$: total cost function
- $\mathbf{s}^*(Z_i)$: optimal solution for task set Z_i

Any grouping of the routes of the optimal solution $\mathbf{s}^*(Z)$ is optimal in terms of Eq. (1)

Obtaining $Z_1, \ldots, Z_g$

- In practice, $\mathbf{s}^*(Z)$ cannot be known
- Approximation: Optimal solution $\rightarrow$ Best-so-far individual
- Grouping the routes of the best-so-far individual $\bar{\mathbf{s}}(Z)$

DECOMPOSITION BASED ON BEST-SO-FAR INDIVIDUAL

Grouping the routes of the best-so-far individual $\bar{\mathbf{s}}(Z)$ can provide a good upper bound of Eq. (1).

$$\sum_{i=1}^{g} tc\left(\mathbf{s}^*(Z_i)\right) \leq \sum_{i=1}^{g} tc\left(\bar{\mathbf{s}}(Z_i)\right) = tc\left(\bar{\mathbf{s}}(Z)\right)$$

As the best-so-far individual $\bar{\mathbf{s}}(Z)$ improves, the upper bound of $\sum_{i=1}^{g} tc\left(\mathbf{s}^*(Z_i)\right)$ also improves, and thus is more likely to be better.

Route Distance Grouping

Domain knowledge

- The total cost is to be minimized
- The tasks that are closer to each other are more likely to be linked together
- The routes that are more tangled with each other are more likely to be in the same subcomponent

Route Distance Grouping

Domain knowledge

- The total cost is to be minimized
- The tasks that are closer to each other are more likely to be linked together
- The routes that are more tangled with each other are more likely to be in the same subcomponent
- Clustering on the route level

Route Distance Grouping

Distance between routes (Tanglement)

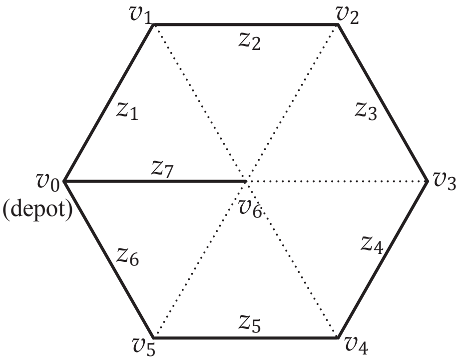
$$\Delta_{\text{task}}(z_1, z_2) = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2}\Delta\big(v_i(z_1), v_j(z_2)\big)}{4} \tag{2}$$

$$\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\sum_{z_1\in\mathbf{s}_1}\sum_{z_2\in\mathbf{s}_2}\Delta_{\text{task}}(z_1, z_2)}{|\mathbf{s}_1|\cdot|\mathbf{s}_2|} \tag{3}$$

$$\hat{\Delta}_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)}{\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_1)}\cdot\frac{\Delta_{\text{route}}(\mathbf{s}_1, \mathbf{s}_2)}{\Delta_{\text{route}}(\mathbf{s}_2, \mathbf{s}_2)} \tag{4}$$

Route Distance Grouping

An example



Route Distance Grouping

Distance between nodes

$$
\Delta = \begin{matrix} & v_0 & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}
\begin{pmatrix}
0 & 1 & 2 & 2 & 2 & 1 & 1 \\
1 & 0 & 1 & 2 & 2 & 2 & 1 \\
2 & 1 & 0 & 1 & 2 & 2 & 1 \\
2 & 2 & 1 & 0 & 1 & 2 & 1 \\
2 & 2 & 2 & 1 & 0 & 1 & 1 \\
1 & 2 & 2 & 2 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0
\end{pmatrix}
$$

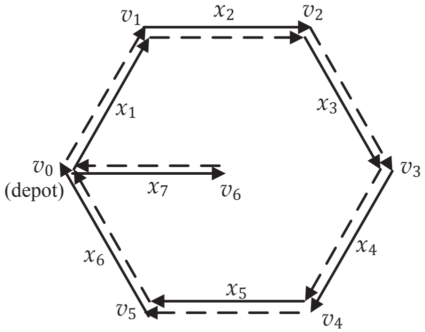
Route Distance Grouping

Distance between tasks

$$\Delta_{\text{task}} = \begin{array}{c} \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{array} \begin{array}{c} \begin{array}{ccccccc} z_1 & z_2 & z_3 & z_4 & z_5 & z_6 & z_7 \end{array} \\ \begin{pmatrix} 1/2 & 1 & 7/4 & 2 & 7/4 & 1 & 3/4 \\ 1 & 1/2 & 1 & 7/4 & 2 & 7/4 & 5/4 \\ 7/4 & 1 & 1/2 & 1 & 7/4 & 2 & 3/2 \\ 2 & 7/4 & 1 & 1/2 & 1 & 7/4 & 3/2 \\ 7/4 & 2 & 7/4 & 1 & 1/2 & 1 & 5/4 \\ 1 & 7/4 & 2 & 7/4 & 1 & 1/2 & 3/4 \\ 3/4 & 5/4 & 3/2 & 3/2 & 5/4 & 3/4 & 1/2 \end{pmatrix} \end{array}$$

Route Distance Grouping

A solution with three routes



$\mathbf{s}_1$=(0,x_1,x_3,x_5,0) $\mathbf{s}_2$=(0,x_2,x_4,x_6,0) $\mathbf{s}_3$=(0,x_7,0)

Route Distance Grouping

Distance between routes

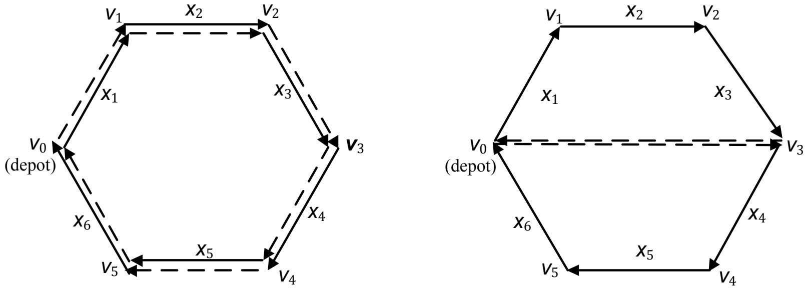
$$\Delta_{\text{route}} = \begin{array}{c} \\ \mathbf{s}_1 \\ \mathbf{s}_2 \\ \mathbf{s}_3 \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \end{array} \\ \begin{pmatrix} 4/3 & 4/3 & 7/6 \\ 4/3 & 4/3 & 7/6 \\ 7/6 & 7/6 & 1/2 \end{pmatrix} \end{array}$$

Normalized distance between routes

$$\hat{\Delta}_{\text{route}} = \begin{array}{c} \\ \mathbf{s}_1 \\ \mathbf{s}_2 \\ \mathbf{s}_3 \end{array} \begin{array}{c} \begin{array}{ccc} \mathbf{s}_1 & \mathbf{s}_2 & \mathbf{s}_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 49/24 \\ 1 & 1 & 49/24 \\ 49/24 & 49/24 & 1 \end{pmatrix} \end{array}$$

Route Distance Grouping

Improved routes



$\mathbf{s}_1=(0,x_1,x_3,x_5,0)$ $\mathbf{s}_2=(0,x_2,x_4,x_6,0)$ $\mathbf{s'}_1=(0,x_1,x_2,x_3,0)$ $\mathbf{s'}_2=(0,x_4,x_5,x_6,0)$

Route Distance Grouping

Improved routes

- $\mathbf{s}_1 = (0, x_1, x_3, x_5, 0)$, $\mathbf{s}_2 = (0, x_2, x_4, x_6, 0)$
- $\mathbf{s}'_1 = (0, x_1, x_2, x_3, 0)$, $\mathbf{s}'_2 = (0, x_4, x_5, x_6, 0)$
- $ts(\mathbf{s}_1) + tc(\mathbf{s}_2) = 6 + 6 = 12$
- $ts(\mathbf{s}'_1) + tc(\mathbf{s}'_2) = 5 + 5 = 10$

Route Distance Grouping

Improved routes

- $\mathbf{s}_1 = (0, x_1, x_3, x_5, 0)$, $\mathbf{s}_2 = (0, x_2, x_4, x_6, 0)$
- $\mathbf{s}'_1 = (0, x_1, x_2, x_3, 0)$, $\mathbf{s}'_2 = (0, x_4, x_5, x_6, 0)$
- $ts(\mathbf{s}_1) + tc(\mathbf{s}_2) = 6 + 6 = 12$
- $ts(\mathbf{s}'_1) + tc(\mathbf{s}'_2) = 5 + 5 = 10$
- There is no improvement when combining $\mathbf{s}_1$ ($\mathbf{s}_2$) with $\mathbf{s}_3$

Route Clustering Problem

Fuzzy k-medoids

Select a subset of (g) routes $\mathbf{c} \subseteq \mathbf{s}$ so that

$$\min_{\mathbf{c} \subseteq \mathbf{s}} \mathcal{J}_\alpha(\mathbf{c}; \mathbf{s}) = \sum_{\mathbf{s}_i \in \mathbf{s} \setminus \mathbf{c}} \sum_{\mathbf{c}_j \in \mathbf{c}} \mathcal{M}_\alpha(\mathbf{s}_i, \mathbf{c}_j) \cdot \hat{\Delta}_{\text{route}}(\mathbf{s}_i, \mathbf{c}_j), \tag{5}$$

where

$$\mathcal{M}_\alpha(\mathbf{s}_i, \mathbf{c}_j) = \frac{\left(\frac{1}{\hat{\Delta}_{\text{route}}(\mathbf{s}_i, \mathbf{c}_j)}\right)^\alpha}{\sum_{k=1}^{g} \left(\frac{1}{\hat{\Delta}_{\text{route}}(\mathbf{s}_i, \mathbf{c}_k)}\right)^\alpha}, \quad \alpha \in [0, \infty) \tag{6}$$

$\alpha = 0 \rightarrow$ Randomly assignment
$\alpha = \infty \rightarrow$ Assign to the closest medoid with no fuzziness

Route Clustering Problem

Finding the medoids

Standard Partitioning Around Medoids algorithm

- Randomly initialize the medoids $\mathbf{c}$
- Repeatedly swap between medoids and non-medoids until no improvement on the objective function is found

Assigning the non-medoids

Roulette wheel based on the membership function $\mathcal{M}_\alpha(\mathbf{s}_i, \mathbf{c}_j)$

Outline

Parameter settings

Optimize each subcomponent Z_i with MAENS (one of the state-of-the-art algorithms for CARP)

Parameter	Description	Value
g	Number of subcomponents	1, 2, 3
α	Fuzziness control parameter	1, 5, 10
$psize$	Population size	30
$offsize$	Offspring population size	$6 \cdot psize$
P_{ls}	Probability of local search	0.2
gen	Maximal generations	500
$cycle$	Number of cycles	50

TEST INSTANCES

The EGL-G large scale CARP test set
τ: minimal number of routes

Name	$\|V\|$	$\|E\|$	$\|Z\|$	τ
EGL-G1-A	255	375	347	20
EGL-G1-B	255	375	347	25
EGL-G1-C	255	375	347	30
EGL-G1-D	255	375	347	35
EGL-G1-E	255	375	347	40
EGL-G2-A	255	375	375	22
EGL-G2-B	255	375	375	27
EGL-G2-C	255	375	375	32
EGL-G2-D	255	375	375	37
EGL-G2-E	255	375	375	42

CONVERGENCE CURVES

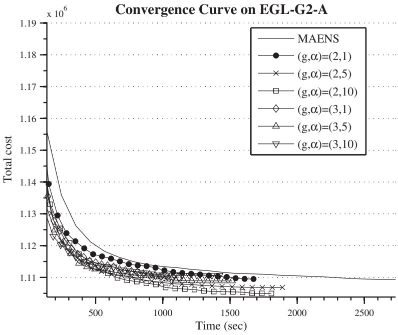


FIGURE: Convergence curves on EGL-G2-A.

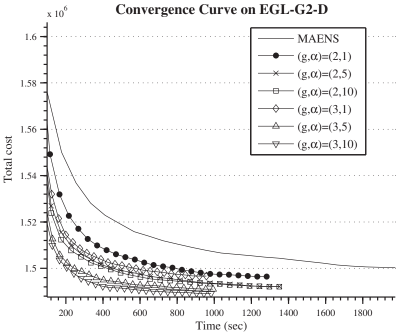


FIGURE: Convergence curves on EGL-G2-D.

Scalability

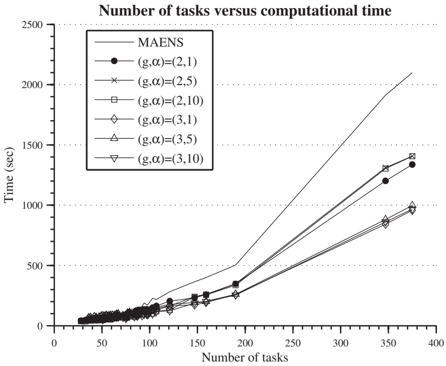


FIGURE: Size versus time for different g and α values.

Analysis on the results

- RDG much improves the quality of solution given the same limited computational resource
- The optimal values of g and α vary from instance to instance
- Scalability is much improved

Outline

Conclusion

- Large Scale CARP (LSCARP) is significant in practice, yet has been overlooked so far
- Cooperative Co-evolution framework is a suitable framework to divide-and-conquer LSCARP
- RDG is an effective strategy to decompose LSCARP
- Both quality of solution and scalability are much improved

Future Work

- Problem size is still not large enough, and the scalability is still quadratic ($O(n^2) \rightarrow O(n^2/g)$): can it be linear?
- Only upper bound of the optimum under the decomposition. Including lower bound can improve the evaluation of decomposition
- Adaptive g and α schemes

Publications

- Yi Mei, Xiaodong Li and Xin Yao, "Decomposing Large-Scale Capacitated Arc Routing Problems using a Random Route Grouping Method," *IEEE Congress on Evolutionary Computation (CEC)*, 2013.
- Yi Mei, Xiaodong Li and Xin Yao, "Cooperative Co-evolution with Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 435-449, 2014.
- Yi Mei, Xiaodong Li and Xin Yao, "Variable Neighborhood Decomposition for Large Scale Capacitated Arc Routing Problem," *IEEE Congress on Evolutionary Computation (CEC)*, 2014.

Thank you!