Cooperative Co-evolution with Online Optimizer Selection for Large-Scale Optimization

Yuan Sun[1,2] Michael Kirley [1] Xiaodong Li [2]

[1]School of Computing and Information Systems, University of Melbourne

[2]School of Science, RMIT University

yuan.sun@unimelb.edu.au
yuan.sun@rmit.edu.au

July 17, 2018

Overview

Introduction: Large-Scale Continuous Optimization

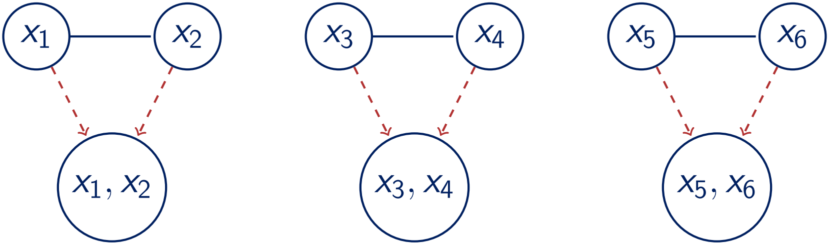
Large-scale (High-dimensional) Continuous Optimization Problems are challenging to solve:

- Search space increases exponentially.
- Problem complexity increases greatly.
- The running time of some evolutionary algorithms increases significantly.

Background: Cooperative Co-evolution (CC)[1]

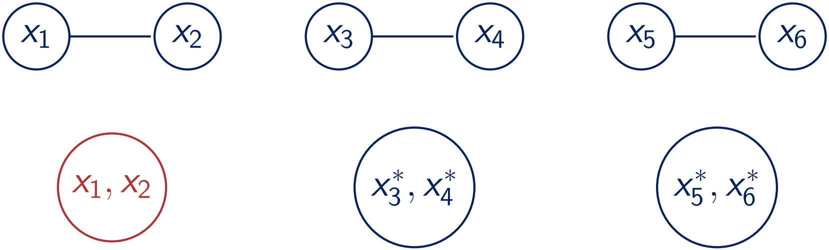
[1] Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Cooperative Co-evolution (CC)[1]



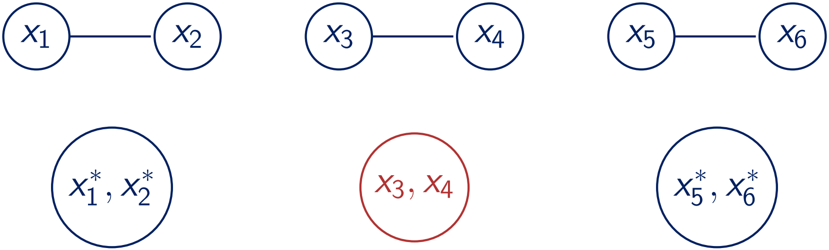
[1] Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Cooperative Co-evolution (CC)[1]



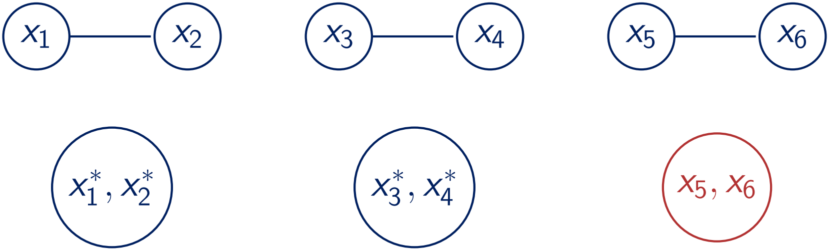
[1] Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Cooperative Co-evolution (CC)[1]



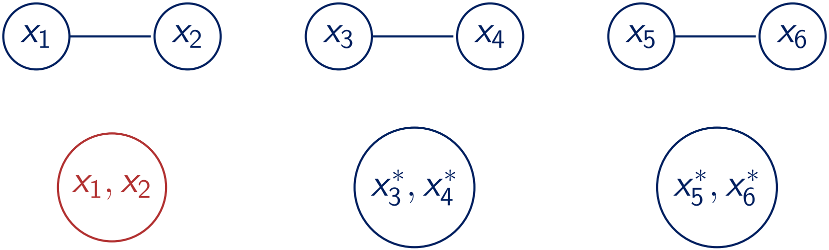
[1] Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Cooperative Co-evolution (CC)[1]



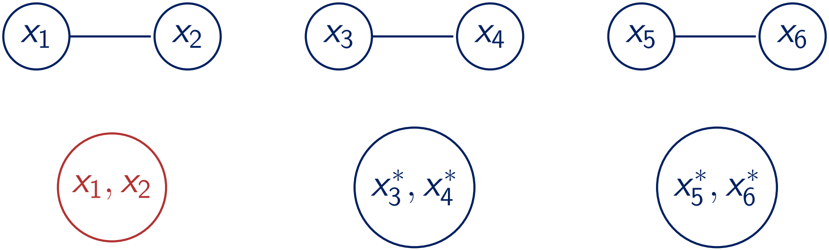
[1]Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Cooperative Co-evolution (CC)[1]



[1] Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Cooperative Co-evolution (CC)[1]



Limitation of CC: Inefficient to solve problems with imbalanced components (contributing differently to the overall fitness values).

[1]Potter M A, De Jong K A. A cooperative coevolutionary approach to function optimization[C]//International Conference on Parallel Problem Solving from Nature. Springer, Berlin, Heidelberg, 1994: 249-257.

Background: Efficient Resource Allocation in CC [2]

Components: x_1, x_2 x_3, x_4 x_5, x_6

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

Background: Efficient Resource Allocation in CC [2]

Components: x_1, x_2 x_3, x_4 x_5, x_6

U_1 :

U_2 :

U_3 :

[2]Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

Background: Efficient Resource Allocation in CC [2]

Components: x_1, x_2 | x_3, x_4 | x_5, x_6

U_1 : 0

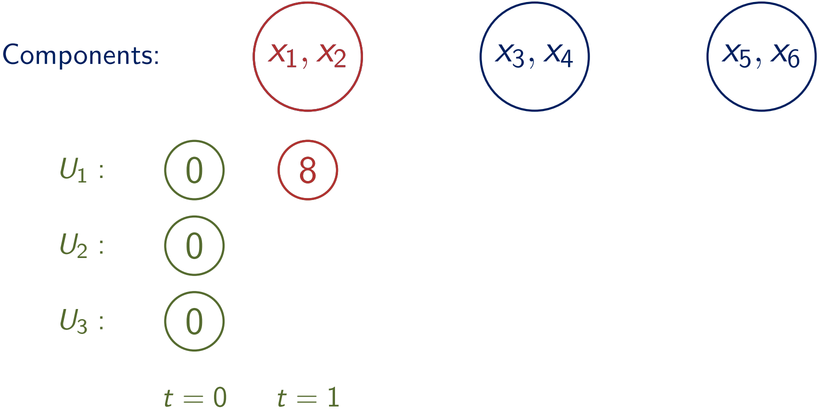
U_2 : 0

U_3 : 0

$t = 0$

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

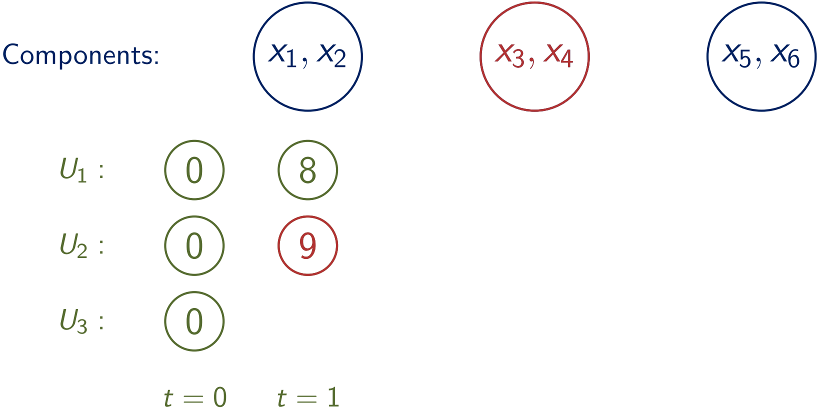
Background: Efficient Resource Allocation in CC [2]



$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \quad (1)$$

[2]Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

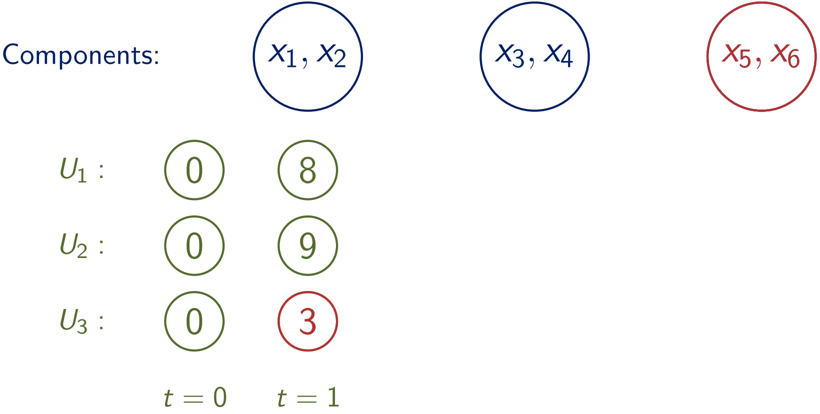
Background: Efficient Resource Allocation in CC [2]



$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \quad (1)$$

[2]Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

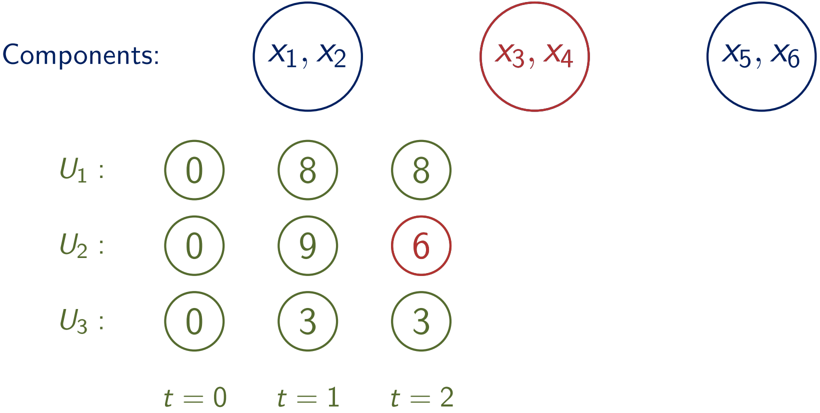
Background: Efficient Resource Allocation in CC [2]



$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \tag{1}$$

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

Background: Efficient Resource Allocation in CC [2]



$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \quad (1)$$

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

Background: Efficient Resource Allocation in CC [2]

Components: x_1, x_2 | x_3, x_4 | x_5, x_6

U_1 :	0	8	8	5
U_2 :	0	9	6	6
U_3 :	0	3	3	3
	$t = 0$	$t = 1$	$t = 2$	$t = 3$

$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \tag{1}$$

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

Background: Efficient Resource Allocation in CC [2]

Components: x_1, x_2 | x_3, x_4 | x_5, x_6

U_1 :	0	8	8	5	5
U_2 :	0	9	6	6	4
U_3 :	0	3	3	3	3
	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$

$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \quad (1)$$

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

Background: Efficient Resource Allocation in CC [2]

Components: x_1, x_2 x_3, x_4 x_5, x_6

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$
U_1 :	0	8	8	5	5	3
U_2 :	0	9	6	6	4	4
U_3 :	0	3	3	3	3	3

$$U_i = (\hat{U}_i + \hat{y}_b - y_b)/2. \quad (1)$$

[2] Yang M, Omidvar M N, Li C, et al. Efficient resource allocation in cooperative co-evolution for large-scale global optimization[J]. IEEE Transactions on Evolutionary Computation, 2017, 21(4): 493-505.

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2

Components: c_1 c_2

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

U_{a_1,c_1} :

U_{a_1,c_2} :

U_{a_2,c_1} :

U_{a_2,c_2} :

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

U_{a_1,c_1} : 0

U_{a_1,c_2} : 0

U_{a_2,c_1} : 0

U_{a_2,c_2} : 0

$t = 0$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

U_{a_1,c_1} : 0 8

U_{a_1,c_2} : 0

U_{a_2,c_1} : 0

U_{a_2,c_2} : 0

$t = 0$ $t = 1$

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

	$t=0$	$t=1$
U_{a_1,c_1} :	0	8
U_{a_1,c_2} :	0	4
U_{a_2,c_1} :	0	
U_{a_2,c_2} :	0	

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

	$t = 0$	$t = 1$
U_{a_1,c_1} :	0	8
U_{a_1,c_2} :	0	4
U_{a_2,c_1} :	0	9
U_{a_2,c_2} :	0	

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

	$t = 0$	$t = 1$
U_{a_1,c_1} :	0	8
U_{a_1,c_2} :	0	4
U_{a_2,c_1} :	0	9
U_{a_2,c_2} :	0	3

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

U_{a_1,c_1} :	0	8	8
U_{a_1,c_2} :	0	4	4
U_{a_2,c_1} :	0	9	6
U_{a_2,c_2} :	0	3	3
	$t = 0$	$t = 1$	$t = 2$

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \tag{2}$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

	$t=0$	$t=1$	$t=2$	$t=3$
U_{a_1,c_1} :	0	8	8	5
U_{a_1,c_2} :	0	4	4	4
U_{a_2,c_1} :	0	9	6	6
U_{a_2,c_2} :	0	3	3	3

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

	$t=0$	$t=1$	$t=2$	$t=3$	$t=4$
U_{a_1,c_1} :	0	8	8	5	5
U_{a_1,c_2} :	0	4	4	4	4
U_{a_2,c_1} :	0	9	6	6	4
U_{a_2,c_2} :	0	3	3	3	3

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Selection Procedures

Algorithms: a_1 a_2 Components: c_1 c_2

U_{a_1,c_1} :	0	8	8	5	5	3
U_{a_1,c_2} :	0	4	4	4	4	4
U_{a_2,c_1} :	0	9	6	6	4	4
U_{a_2,c_2} :	0	3	3	3	3	3
	$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$

$$U_{a_i,c_j} = \frac{\hat{U}_{a_i,c_j} + (\hat{y}_b - y_b)/\hat{y}_b}{2}. \quad (2)$$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$

$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$

$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$

$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$

$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$

$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$

$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i + 1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i+1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

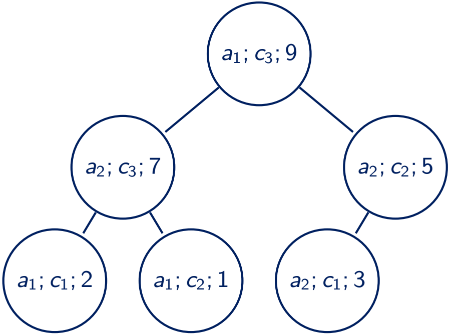
2. using a max-heap: $O\big(T \log(|\mathbb{A}||\mathbb{C}|)\big)$.

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i + 1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

2. using a max-heap: $O\big(T \log(|\mathbb{A}||\mathbb{C}|)\big)$.



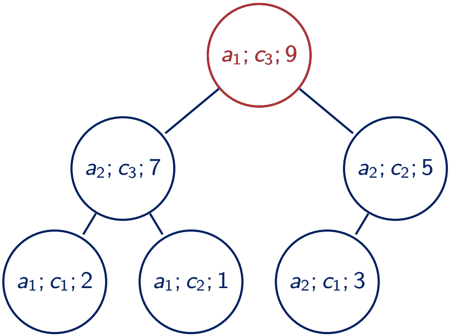
$t = i$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i + 1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

2. using a max-heap: $O\big(T\log(|\mathbb{A}||\mathbb{C}|)\big)$.



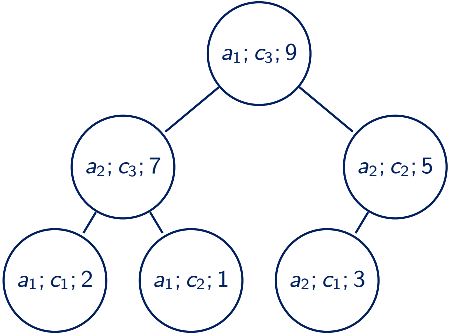
$t = i$

CC with Optimizer Selection: Time Complexity

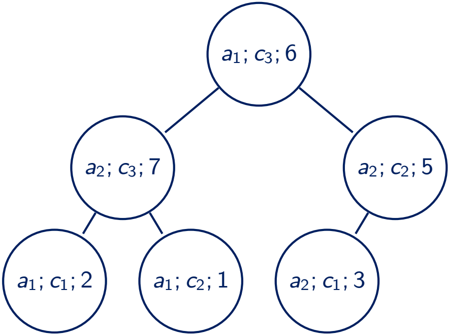
1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i + 1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

2. using a max-heap: $O\big(T\log(|\mathbb{A}||\mathbb{C}|)\big)$.



$t = i$



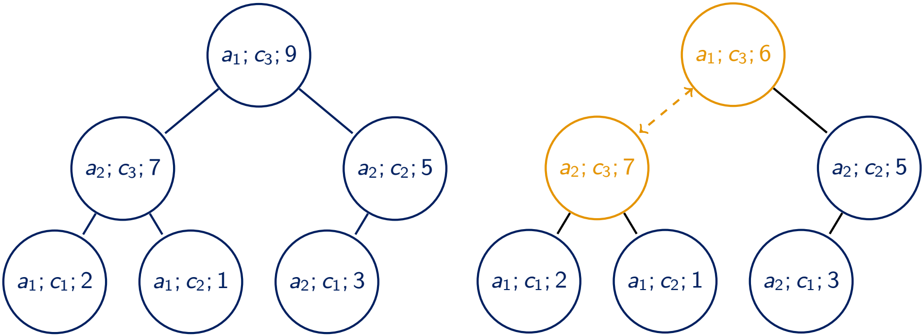
$t = i + 1$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i + 1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

2. using a max-heap: $O\big(T\log(|\mathbb{A}||\mathbb{C}|)\big)$.



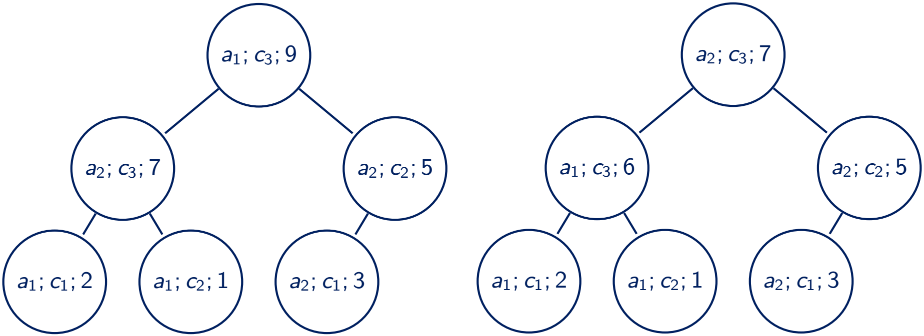
$t = i$ $\qquad$ $t = i + 1$

CC with Optimizer Selection: Time Complexity

1. using an array: $\Theta(|\mathbb{A}||\mathbb{C}|T)$.

$t = i$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 9$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$
$t = i + 1$	$a_1; c_1; 2$	$a_1; c_2; 1$	$a_1; c_3; 6$	$a_2; c_1; 3$	$a_2; c_2; 5$	$a_2; c_3; 7$

2. using a max-heap: $O\big(T \log(|\mathbb{A}||\mathbb{C}|)\big)$.



$t = i$ $\qquad$ $t = i + 1$

Experimental Methodology

- Decomposition method: Recursive Differential Grouping[3] which uses $O(n\log(n))$ function evaluations in decomposition.

[3] Sun Y, Kirley M, Halgamuge S K. A recursive decomposition method for large scale continuous optimization[J]. IEEE Transactions on Evolutionary Computation, accepted November 2017.

Experimental Methodology

- Decomposition method: Recursive Differential Grouping[3] which uses $O(n\log(n))$ function evaluations in decomposition.
- Benchmark problems: CEC'2010 benchmark large-scale global optimization problems.

[3] Sun Y, Kirley M, Halgamuge S K. A recursive decomposition method for large scale continuous optimization[J]. IEEE Transactions on Evolutionary Computation, accepted November 2017.

Experimental Methodology

- Decomposition method: Recursive Differential Grouping[3] which uses $O(n\log(n))$ function evaluations in decomposition.
- Benchmark problems: CEC'2010 benchmark large-scale global optimization problems.
- CCOS: uses Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) and Social-Learning Particle Swarm Optimization (SL-PSO) as the candidate optimizers.

[3] Sun Y, Kirley M, Halgamuge S K. A recursive decomposition method for large scale continuous optimization[J]. IEEE Transactions on Evolutionary Computation, accepted November 2017.

Experimental Methodology

- Decomposition method: Recursive Differential Grouping[3] which uses $O(n\log(n))$ function evaluations in decomposition.
- Benchmark problems: CEC'2010 benchmark large-scale global optimization problems.
- CCOS: uses Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) and Social-Learning Particle Swarm Optimization (SL-PSO) as the candidate optimizers.
- CCDE: only uses SaNSDE as the optimizer.

[3] Sun Y, Kirley M, Halgamuge S K. A recursive decomposition method for large scale continuous optimization[J]. IEEE Transactions on Evolutionary Computation, accepted November 2017.

Experimental Methodology

- Decomposition method: Recursive Differential Grouping[3] which uses $O(n \log(n))$ function evaluations in decomposition.
- Benchmark problems: CEC'2010 benchmark large-scale global optimization problems.
- CCOS: uses Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) and Social-Learning Particle Swarm Optimization (SL-PSO) as the candidate optimizers.
- CCDE: only uses SaNSDE as the optimizer.
- CCPSO: only uses SL-PSO as the optimizer.

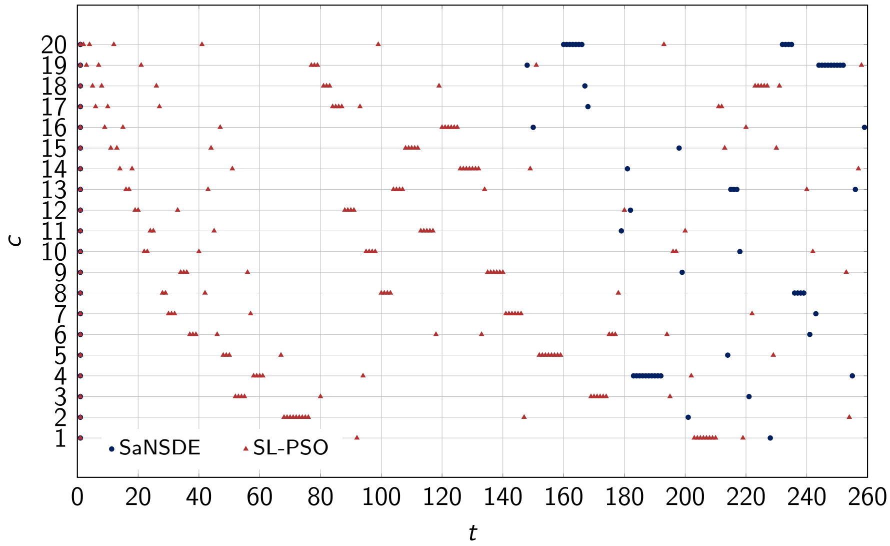
[3] Sun Y, Kirley M, Halgamuge S K. A recursive decomposition method for large scale continuous optimization[J]. IEEE Transactions on Evolutionary Computation, accepted November 2017.

Experimental Methodology

- Decomposition method: Recursive Differential Grouping[3] which uses $O(n\log(n))$ function evaluations in decomposition.
- Benchmark problems: CEC'2010 benchmark large-scale global optimization problems.
- CCOS: uses Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) and Social-Learning Particle Swarm Optimization (SL-PSO) as the candidate optimizers.
- CCDE: only uses SaNSDE as the optimizer.
- CCPSO: only uses SL-PSO as the optimizer.
- Statistical test: Wilcoxon rank-sum test with 95% confidence interval.

[3] Sun Y, Kirley M, Halgamuge S K. A recursive decomposition method for large scale continuous optimization[J]. IEEE Transactions on Evolutionary Computation, accepted November 2017.

Experimental Results: Selection Details of CCOS (f_{16})



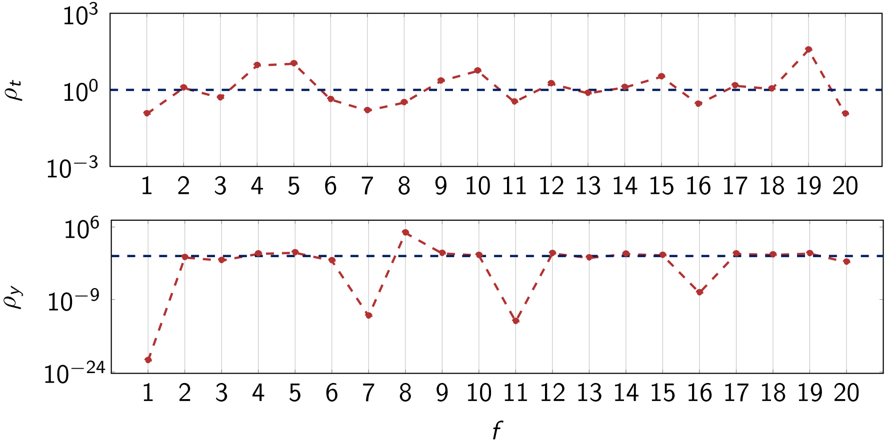
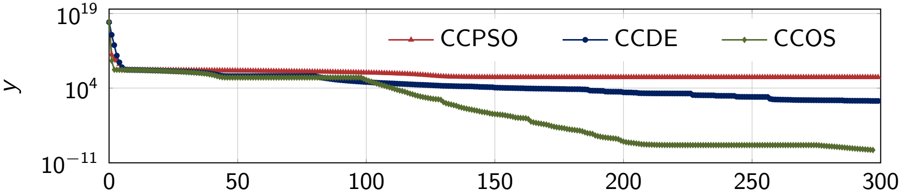
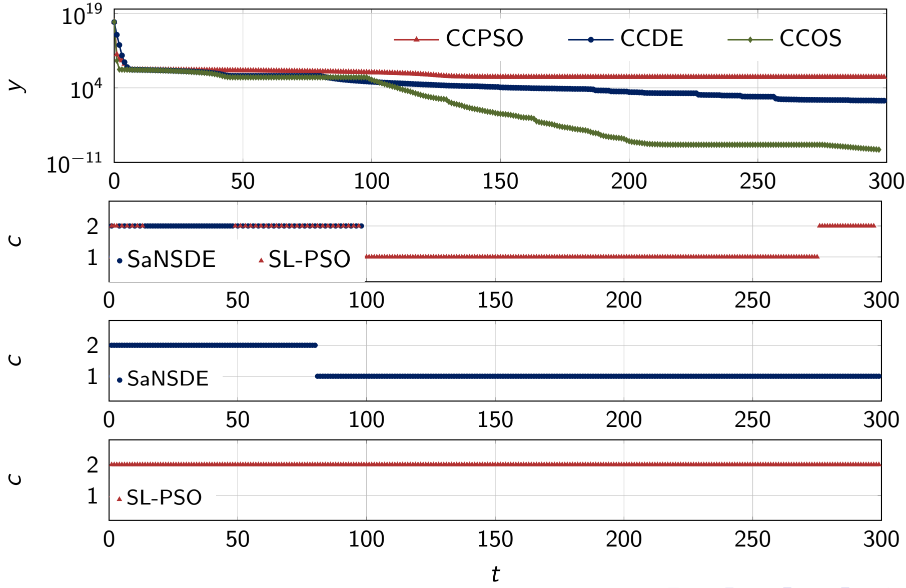


Figure: ρ_t: the ratio of the number of evolutionary cycles that DE and PSO were selected; ρ_y: the ratio of the solution quality generated by CCPSO and CCDE.

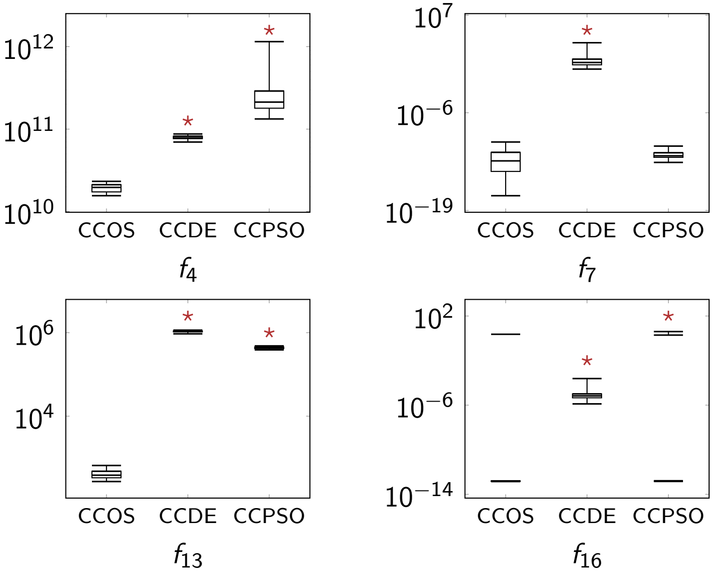
Experimental Results: Selection Ability of CCOS (f_8)



Experimental Results: Selection Ability of CCOS (f_8)



Experimental Results: Comparison with CCDE and CCPSO



Experimental Results: Comparison with State-of-the-arts

Table: The optimization results of CCOS, CSO, MOS and MA-SW-Chain when used to solve the CEC'2010 benchmark problems (Wilcoxon rank-sum tests).

Func	Stats	CCOS	CSO	MOS	MA-SW-Chain
f_4	median	1.96e+10	7.26e+11	4.94e+11	3.10e+11
	mean	1.95e+10	7.25e+11	5.16e+11	2.97e+11
	std	2.23e+09	1.23e+11	1.85e+11	6.19e+10
f_7	median	3.94e-13	2.04e+04	2.27e+07	7.94e-03
	mean	9.15e-12	2.01e+04	3.54e+07	1.17e+02
	std	2.61e-11	3.86e+03	3.22e+07	2.37e+02
f_{13}	median	3.85e+02	5.47e+02	3.19e+02	8.61e+02
	mean	4.10e+02	6.29e+02	3.32e+02	9.83e+02
	std	1.09e+02	2.32e+02	1.19e+02	5.66e+02
f_{16}	median	1.49e-13	5.75e-08	3.97e+02	9.32e+01
	mean	1.88e-01	5.89e-08	3.96e+02	9.95e+01
	std	5.53e-01	5.61e-09	3.47e+00	1.53e+01

Conclusion

1. Proposed an online optimizer selection framework to select the best optimizer from a portfolio for each component when solving large-scale optimization problems using CC algorithm.

Conclusion

1. Proposed an online optimizer selection framework to select the best optimizer from a portfolio for each component when solving large-scale optimization problems using CC algorithm.
2. Experimentally demonstrated that the proposed CCOS algorithm was successful in selecting the best optimizer when solving the CEC'2010 benchmark problems.

Conclusion

1. Proposed an online optimizer selection framework to select the best optimizer from a portfolio for each component when solving large-scale optimization problems using CC algorithm.
2. Experimentally demonstrated that the proposed CCOS algorithm was successful in selecting the best optimizer when solving the CEC'2010 benchmark problems.
3. Showed that CCOS could potentially generate statistically better solution quality than the default CC algorithm with no optimizer selection ability.

Thank You! & Questions?