Improving Local Search Algorithms via Probabilistic Configuration Checking

Weilin Luo [1], Rongzhen Ye [1], Hai Wan [1,*], Shaowei Cai [2,*],
Biqing Fang [1] Delong Zhang [1]

[1] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China

HCP Report of AAAI 2022 Paper

Content

Content

Definition of Problem

Combinatorial optimization problems (COPs)

- Minimum weight dominating set problem (MWDS)

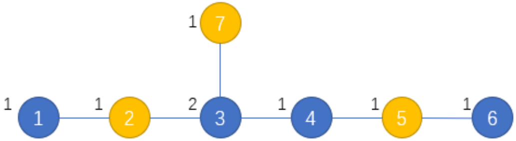


Figure 1: Example for MWDS.

Definition of Problem

Combinatorial optimization problems (COPs)

- Minimum weight dominating set problem (MWDS)

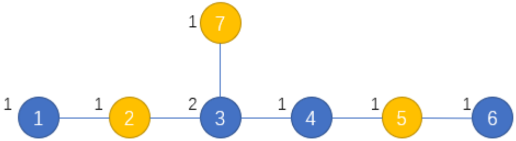


Figure 1: Example for MWDS.

- Maximum weight clique problem (MWCP)

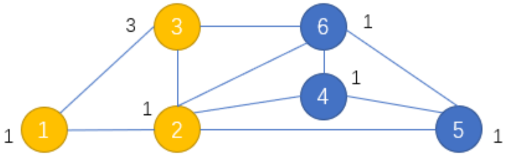


Figure 2: Example for MWCP.

Definition of Problem

Local search

- A popular method for solving COPs [4,9–11]

Definition of Problem

Local search

- A popular method for solving COPs[4,9–11]
- Configuration checking (CC)[3]: alleviate the *cycling problem* for COPs

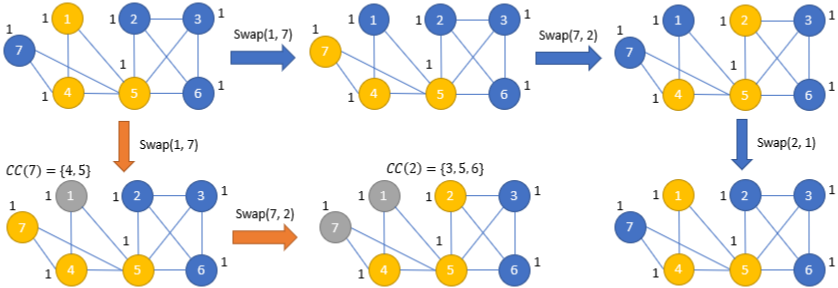


Figure 3: CC for MWCP.

Related Work

The variants of CC

- Redefinition of configuration (What we focus)
  - 1-level neighborhoods: [1,2,10]
  - 2-level neighborhoods: [8,11]
- Multi-value CC
- Changes of CC rules

Related Work

The variants of CC

- Redefinition of configuration (What we focus)
  - 1-level neighborhoods: [1,2,10]
  - 2-level neighborhoods: [8,11]
- Multi-value CC
- Changes of CC rules

CC strategy of CC^2FS[11] (MWDS 2-level neighborhoods)

- When removing a vertex u from the candidate solution S
  - $confCh[u] := 0$;
  - for each vertex $w \in N_1(u) \cup N_2(u)$, $confCh[w]$ is set to 1.
- When adding a vertex v to the candidate solution S
  - for each vertex $w \in N_1(u) \cup N_2(u)$, $confCh[w]$ is set to 1.

Related Work

The variants of CC

- Redefinition of configuration (What we focus)
  - 1-level neighborhoods: [1,2,10]
  - 2-level neighborhoods: [8,11]
- Multi-value CC
- Changes of CC rules

CC strategy of CC^2FS[11] (MWDS 2-level neighborhoods)

- When removing a vertex u from the candidate solution S
  - $confCh[u] := 0$;
  - for each vertex $w \in N_1(u) \cup N_2(u)$, $confCh[w]$ is set to 1.
- When adding a vertex v to the candidate solution S
  - for each vertex $w \in N_1(u) \cup N_2(u)$, $confCh[w]$ is set to 1.

What about 3-, 4-, or larger-level neighborhoods?

Motivating Example

Neighborhoods with more than two levels can have an impact on a vertex and should be considered in the configuration.

Motivating Example

Neighborhoods with more than two levels can have an impact on a vertex and should be considered in the configuration.

CC^2FS-H: variants of CC^2FS[11] (2-level neighborhoods)

- When removing a vertex u from the candidate solution S
  - $confCh[u] := 0$;
  - randomly choose half neighborhoods as configuration
  - for each vertex w in the configuration, $confCh[w]$ is set to 1.
- When adding a vertex v to the candidate solution S
  - randomly choose half neighborhoods as configuration
  - for each vertex w in the configuration, $confCh[w]$ is set to 1.

Motivating Example

Neighborhoods with more than two levels can have an impact on a vertex and should be considered in the configuration.

CC^{2}FS-H: variants of CC^{2}FS [11] (2-level neighborhoods)

- When removing a vertex u from the candidate solution S
  - $confCh[u] := 0$;
  - randomly choose half neighborhoods as configuration
  - for each vertex w in the configuration, $confCh[w]$ is set to 1.
- When adding a vertex v to the candidate solution S
  - randomly choose half neighborhoods as configuration
  - for each vertex w in the configuration, $confCh[w]$ is set to 1.

Table 1: CC^{2}FS vs. CC^{2}FS-H.

Instances	CC^{2}FS			CC^{2}FS-H		
	min	avg	time	min	avg	time (s)
frb30-15-1	212	214.2	2.59	212	**212.0**	**0.11**
frb30-15-2	242	242.0	8.81	242	242.0	**0.18**
frb30-15-3	175	175.0	**0.01**	175	175.0	0.03
frb30-15-4	166	168.6	0.19	166	**166.0**	**0.08**
frb30-15-5	160	160.0	**0.01**	160	160.0	0.02
frb35-17-1	274	274.9	0.57	274	**274.0**	**0.11**
frb35-17-2	208	208.3	67.44	208	**208.0**	**0.29**
frb35-17-3	201	201.0	0.38	201	201.0	**0.10**
frb35-17-4	286	286.8	0.73	286	**286.0**	**0.45**
frb35-17-5	295	295.7	5.60	295	**295.0**	**0.11**
frb40-19-1	262	262.0	1.99	262	262.0	**0.14**
frb40-19-2	243	243.9	0.66	243	**243.0**	**0.18**
frb40-19-3	250	251.8	102.43	250	**250.0**	**3.88**
frb40-19-4	250	250.0	0.07	**249**	**249.0**	95.82
frb40-19-5	272	282.3	0.31	272	**277.6**	82.50
frb45-21-1	328	328.3	7.82	328	**328.0**	**0.64**
frb45-21-2	259	261.7	46.82	259	**259.1**	**1.39**
frb45-21-3	233	233.9	9.19	233	**233.0**	**0.89**
frb45-21-4	399	399.2	22.67	399	**399.0**	**0.95**
frb45-21-5	312	318.8	128.19	312	**312.0**	**1.82**
frb50-23-1	261	264.7	90.71	261	**261.0**	**8.97**
frb50-23-2	277	277.0	**0.01**	277	277.0	0.17
frb50-23-3	299	301.9	1.21	**281**	**292.2**	31.13
frb50-23-4	265	265.0	3.04	265	265.0	**0.47**
frb50-23-5	410	418.8	128.09	410	**411.1**	158.96
frb53-24-1	229	230.0	95.41	229	**229.0**	**2.35**
frb53-24-2	298	298.0	1.04	298	298.0	**0.38**
frb53-24-3	182	182.1	0.01	182	**182.0**	0.26
frb53-24-4	189	189.0	**0.02**	189	189.0	0.27
frb53-24-5	204	204.0	**0.01**	204	204.0	0.29
frb56-25-1	229	229.2	0.32	229	**229.0**	0.41
frb56-25-2	319	319.5	25.27	319	**319.0**	**3.46**
frb56-25-3	336	338.7	30.21	336	**336.0**	38.80
frb56-25-4	268	268.0	**0.21**	268	268.0	0.41
frb56-25-5	425	427.6	103.13	425	**425.0**	**1.83**
frb59-26-1	262	264.0	1.62	262	**262.0**	25.31
frb59-26-2	383	391.1	97.04	383	**383.0**	**5.64**
frb59-26-3	248	248.0	1.13	**246**	**246.0**	14.24
frb59-26-4	248	248.9	47.47	248	**248.0**	**0.62**
frb59-26-5	288	288.8	195.62	288	**288.0**	**3.26**
frb100-40	350	350.0	**2.04**	350	350.0	9.70
#win	0	0	7	**3**	**29**	**25**

Motivating Example

Neighborhoods with more than two levels can have an impact on a vertex and should be considered in the configuration.

CC^{2}FS-H: variants of CC^{2}FS[11] (2-level neighborhoods)

- When removing a vertex u from the candidate solution S
  - $confCh[u] := 0$;
  - randomly choose half neighborhoods as configuration
  - for each vertex w in the configuration, $confCh[w]$ is set to 1.
- When adding a vertex v to the candidate solution S
  - randomly choose half neighborhoods as configuration
  - for each vertex w in the configuration, $confCh[w]$ is set to 1.

1. Is there a general way to define the configuration of a vertex?
2. How does the new configuration lead to a further improvement of the CC strategy?

Table 1: CC^{2}FS vs. CC^{2}FS-H.

Instances	CC^{2}FS			CC^{2}FS-H		
	min	avg	time	min	avg	time (s)
frb30-15-1	212	214.2	2.59	212	**212.0**	**0.11**
frb30-15-2	242	242.0	8.81	242	242.0	**0.18**
frb30-15-3	175	175.0	**0.01**	175	175.0	0.03
frb30-15-4	166	168.6	0.19	166	**166.0**	**0.08**
frb30-15-5	160	160.0	**0.01**	160	160.0	0.02
frb35-17-1	274	274.9	0.57	274	**274.0**	**0.11**
frb35-17-2	208	208.3	67.44	208	**208.0**	**0.29**
frb35-17-3	201	201.0	0.38	201	201.0	**0.10**
frb35-17-4	286	286.8	0.73	286	**286.0**	**0.45**
frb35-17-5	295	295.7	5.60	295	**295.0**	**0.11**
frb40-19-1	262	262.0	1.99	262	262.0	**0.14**
frb40-19-2	243	243.9	0.66	243	**243.0**	**0.18**
frb40-19-3	250	251.8	102.43	250	**250.0**	**3.88**
frb40-19-4	250	250.0	0.07	**249**	**249.0**	95.82
frb40-19-5	272	282.3	0.31	272	**277.6**	82.50
frb45-21-1	328	328.3	7.82	328	**328.0**	**0.64**
frb45-21-2	259	261.7	46.82	259	**259.1**	**1.39**
frb45-21-3	233	233.9	9.19	233	**233.0**	**0.89**
frb45-21-4	399	399.2	22.67	399	**399.0**	**0.95**
frb45-21-5	312	318.8	128.19	312	**312.0**	**1.82**
frb50-23-1	261	264.7	90.71	261	**261.0**	**8.97**
frb50-23-2	277	277.0	**0.01**	277	277.0	0.17
frb50-23-3	299	301.9	1.21	**281**	**292.2**	31.13
frb50-23-4	265	265.0	3.04	265	265.0	**0.47**
frb50-23-5	410	418.8	128.09	410	**411.1**	158.96
frb53-24-1	229	230.0	95.41	229	**229.0**	**2.35**
frb53-24-2	298	298.0	1.04	298	298.0	**0.38**
frb53-24-3	182	182.1	0.01	182	**182.0**	0.26
frb53-24-4	189	189.0	**0.02**	189	189.0	0.27
frb53-24-5	204	204.0	**0.01**	204	204.0	0.29
frb56-25-1	229	229.2	0.32	229	**229.0**	0.41
frb56-25-2	319	319.5	25.27	319	**319.0**	**3.46**
frb56-25-3	336	338.7	30.21	336	**336.0**	38.80
frb56-25-4	268	268.0	**0.21**	268	268.0	0.41
frb56-25-5	425	427.6	103.13	425	**425.0**	**1.83**
frb59-26-1	262	264.0	1.62	262	**262.0**	25.31
frb59-26-2	383	391.1	97.04	383	**383.0**	**5.64**
frb59-26-3	248	248.0	1.13	**246**	**246.0**	14.24
frb59-26-4	248	248.9	47.47	248	**248.0**	**0.62**
frb59-26-5	288	288.8	195.62	288	**288.0**	**3.26**
frb100-40	350	350.0	**2.04**	350	350.0	9.70
#win	0	0	7	**3**	**29**	**25**

Content

Preliminaries

- An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$.

Preliminaries

- An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$.
- For an edge $e = \{u, v\}$, vertices u and v are the endpoints of edge e.

Preliminaries

- An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$.
- For an edge $e = \{u, v\}$, vertices u and v are the endpoints of edge e.
- Two vertices are *neighboring vertices* if and only if they belong to one edge.

Preliminaries

- An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$.
- For an edge $e = \{u, v\}$, vertices u and v are the endpoints of edge e.
- Two vertices are *neighboring vertices* if and only if they belong to one edge.
- For a vertex v, the set of its neighboring vertices is $N(v) = \{u \in V \mid \{u, v\} \in E\}$, and its *degree* is $deg(v) = |N(v)|$.

Preliminaries

- An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$.
- For an edge $e = \{u, v\}$, vertices u and v are the endpoints of edge e.
- Two vertices are *neighboring vertices* if and only if they belong to one edge.
- For a vertex v, the set of its neighboring vertices is $N(v) = \{u \in V \mid \{u, v\} \in E\}$, and its *degree* is $deg(v) = |N(v)|$.
- The distance between two vertices u and v, denoted by $dist(u, v)$, is the number of edges in the shortest path between them.

Preliminaries

- An undirected graph $G = (V, E)$ consists of a vertex set V and an edge set $E \subseteq V \times V$.
- For an edge $e = \{u, v\}$, vertices u and v are the endpoints of edge e.
- Two vertices are *neighboring vertices* if and only if they belong to one edge.
- For a vertex v, the set of its neighboring vertices is $N(v) = \{u \in V \mid \{u, v\} \in E\}$, and its *degree* is $deg(v) = |N(v)|$.
- The distance between two vertices u and v, denoted by $dist(u, v)$, is the number of edges in the shortest path between them.

Definition 1 (i-th-level Neighborhoods)

For a vertex v, we define its i-th-level neighborhoods as $N_i(v) = \{u \in V \mid dist(u, v) = i\}$, i-level neighborhoods as $N_{\leq i}(v) = \bigcup_{j=1}^{i} N_j(v)$, and its i^+-level neighborhoods as $N_{>i}(v) = \bigcup_{j=i+1} N_j(v)$, particularly, $N_1(v) = N_{\leq 1}(v) = N(v)$.

Probabilistic Configuration

Definition 2 (Probabilistic Configuration (PC))

Given an undirected graph $G = (V, E)$ and a candidate solution S, the probabilistic configuration (PC) of a vertex $v \in V$ is a vector consisting of the states of all vertices in $(\bigcup_{i=1}^{2}(N_i(v))^{p_i}) \cup (N_{>2}(v))^{p_3}$, where $p_i \in [0, 1]$, $i = 1, 2, 3$, and $\{*\}^{p_i}$ is randomly chosen vertices from the set $\{*\}$ with the size of $\lceil |\{*\}| \cdot p_i \rceil$.

Probabilistic Configuration

Definition 2 (Probabilistic Configuration (PC))

Given an undirected graph $G = (V, E)$ and a candidate solution S, the probabilistic configuration (PC) of a vertex $v \in V$ is a vector consisting of the states of all vertices in $(\bigcup_{i=1}^{2}(N_i(v))^{p_i}) \cup (N_{>2}(v))^{p_3}$, where $p_i \in [0, 1]$, $i = 1, 2, 3$, and $\{*\}^{p_i}$ is randomly chosen vertices from the set $\{*\}$ with the size of $\lceil |\{*\}| \cdot p_i \rceil$.

Existing configurations are *special cases* of the PC.

Table 2: Configurations in the form of PC.

Configuration	Probabilistic configuration form
$N_1(v)$	$(N_1(v))^{1.0} \cup (N_2(v))^{0.0} \cup (N_{>2}(v))^{0.0}$
$N_1(v) \cup N_2(v)$	$\bigcup_{i=1}^{2}(N_i(v))^{1.0} \cup (N_{>2}(v))^{0.0}$

Probabilistic Configuration Checking

Probabilistic Configuration Checking (PCC)

1. At beginning, compute the 1st-, 2nd-, and 2^+-level neighborhoods for each vertex
   - quickly implemented with a breadth-first search algorithm $(O(|V| \cdot |E|))$
   - only needs to be run once

Probabilistic Configuration Checking

Probabilistic Configuration Checking (PCC)

1. At beginning, compute the 1st-, 2nd-, and 2^+-level neighborhoods for each vertex
   - quickly implemented with a breadth-first search algorithm ($O(|V| \cdot |E|)$)
   - only needs to be run once
2. In each search step, for the operated vertex v, select vertices from $N_1(v)$, $N_2(v)$ and $N_{>2}(v)$ to form its configuration

Algorithm 2 SelectConf(v, $N_1(v)$, $N_2(v)$, $N_{>2}(v)$, P)

Input: a vertex v and $N_1(v)$, $N_2(v)$, $N_{>2}(v)$, $P = \{p_1, p_2, p_3\}$ a set of probabilities.
Output: the configuration $Conf$.

1: $Q_1 \leftarrow$ randomly choose $\lceil |N_1(v)| \cdot p_1 \rceil$ vertices from $N_1(v)$;
2: $Q_2 \leftarrow$ randomly choose $\lceil |N_2(v)| \cdot p_2 \rceil$ vertices from $N_2(v)$;
3: $Q_3 \leftarrow$ randomly choose $\lceil |N_{>2}(v)| \cdot p_3 \rceil$ vertices from $N_{>2}(v)$;
4: $Conf(v) \leftarrow Q_1 \cup Q_2 \cup Q_3$;
5: **return** $Conf(v)$;

Example of PC and PCC

Example 1

PC: $p_1 = p_2 = p_3 = 0.5$
For $v = 3$, $N_1(3) = \{2, 4, 7\}$, $N_2(3) = \{1, 5\}$, $N_{>2}(3) = \{6\}$.
Calculate $Conf[3]$ according to Algorithm 1.
Assume $Q_1 \leftarrow \{4, 7\}$, $Q_2 \leftarrow \{1\}$, $Q_3 \leftarrow \{6\}$.
$Conf[3] \leftarrow Q_1 \cup Q_2 \cup Q_3 = \{1, 4, 6, 7\}$.

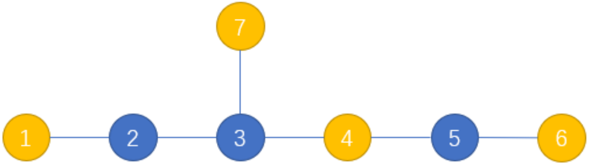


Figure 4: Example for vertex 3.

A PCC Strategy for MWDS

When removing a vertex u from the candidate solution S

- $confCh[u] := 0$;
- calculate $Conf[u]$ according to Algorithm 1;
- for each vertex $w \in Conf[u]$, $confCh[w]$ is set to 1.

When adding a vertex v to the candidate solution S

- calculate $Conf[v]$ according to Algorithm 1;
- for each vertex $w \in Conf[v]$, $confCh[w]$ is set to 1.

A PCC Strategy for MWCP

When removing a vertex u from the candidate solution S

- $confCh[u] := 0;$

When adding a vertex v to the candidate solution S

- calculate $Conf[v]$ according to Algorithm 1;
- for each vertex $w \in Conf[v]$, $confCh[w]$ is set to 1.

When swapping a vertex u from the candidate solution S with a vertex v

- $confCh[u] := 0;$
- calculate $Conf[v]$ according to Algorithm 1;
- for each vertex $w \in Conf[v]$, $confCh[w]$ is set to 1.

Complexity Analysis

- Given an undirected graph $G = (V, E)$, we use $\Delta(G)$ to denote $max\{|deg(v)| \mid v \in V\}$.

Complexity Analysis

- Given an undirected graph $G = (V, E)$, we use $\Delta(G)$ to denote $max\{|deg(v)| \mid v \in V\}$.
- For the updating rules of the 1-level neighborhoods configuration, the worst time complexity is $O(\Delta(G))$.

Complexity Analysis

- Given an undirected graph $G = (V, E)$, we use $\Delta(G)$ to denote $max\{|deg(v)| \mid v \in V\}$.
- For the updating rules of the 1-level neighborhoods configuration, the worst time complexity is $O(\Delta(G))$.
- As for the rules of the 2-level neighborhoods configuration, the worst time complexity is $O(\Delta(G)^2)$.

Complexity Analysis

- Given an undirected graph $G = (V, E)$, we use $\Delta(G)$ to denote $max\{|deg(v)| \mid v \in V\}$.
- For the updating rules of the 1-level neighborhoods configuration, the worst time complexity is $O(\Delta(G))$.
- As for the rules of the 2-level neighborhoods configuration, the worst time complexity is $O(\Delta(G)^2)$.
- For PC, the worst time complexity is $O(|V|)$ ($p_i = 1.0$ for all $1 \leq i \leq 3$)
  - Time-consuming
  - Fortunately, p_i is not 1.0 after a training process in most cases

Complexity Analysis

- Given an undirected graph $G = (V, E)$, we use $\Delta(G)$ to denote $max\{|deg(v)| \mid v \in V\}$.
- For the updating rules of the 1-level neighborhoods configuration, the worst time complexity is $O(\Delta(G))$.
- As for the rules of the 2-level neighborhoods configuration, the worst time complexity is $O(\Delta(G)^2)$.
- For PC, the worst time complexity is $O(|V|)$ ($p_i = 1.0$ for all $1 \leq i \leq 3$)
  - Time-consuming
  - Fortunately, p_i is not 1.0 after a training process in most cases

The complexity of PCC is roughly comparable with CC.

Automatic Configuration Process

Three parameters in the definition of PC: p_1, p_2, and p_3

- configurator: like SMAC[5]

Automatic Configuration Process

Three parameters in the definition of PC: p_1, p_2, and p_3

- configurator: like SMAC [5]

Training set

COP	Benchmarks	Instances	t.t.(s)
MWDS	T1	1000_1000_1, 1000_5000_3, 1000_10000_6, 1000_15000_6, 1000_20000_1	2
	T2	1000_1000_4, 1000_5000_6, 1000_10000_2, 1000_15000_6, 1000_20000_1	2
	UDG	S3N800U150, S0N800U200, S7N1000U150, S6N1000U200	10
	BHSOLIB	50-23-1.mis, 53-24-3.mis, 56-25-3.mis, 59-26-4.mis	30
	DIMACS	brock800_4.mis, C4000.5.mis, DSJC1000.5.mis, gen400_p0.9_75.mis, hamming10-4.mis, keller6.mis, MANN_a81.mis, p_hat1500-3.mis	10
MWCP	BHSOLIB	50-23-5.clq, 53-24-4.clq, 56-25-3.clq, 59-26-1.clq	30
	DIMACS	brock800_4.clq, C4000.5.clq, DSJC1000.5.clq, gen400_p0.9_75.clq, hamming10-4.clq, keller6.clq, MANN_a81.clq, p_hat1500-3.clq	10

Table 3: The detail about the training set, where 't.t.' means the evaluation time of each instance during training.

Automatic Configuration Process

Three parameters in the definition of PC: p_1, p_2, and p_3

- configurator: like SMAC [5]

Training set

COP	Benchmarks	Instances	t.t.(s)
MWDS	T1	1000_1000_1, 1000_5000_3, 1000_10000_6, 1000_15000_6, 1000_20000_1	2
MWDS	T2	1000_1000_4, 1000_5000_6, 1000_10000_2, 1000_15000_6, 1000_20000_1	2
MWDS	UDG	S3N800U150, S0N800U200, S7N1000U150, S6N1000U200	10
MWDS	BHSOLIB	50-23-1.mis, 53-24-3.mis, 56-25-3.mis, 59-26-4.mis	30
MWDS	DIMACS	brock800_4.mis, C4000.5.mis, DSJC1000.5.mis, gen400_p0.9_75.mis, hamming10-4.mis, keller6.mis, MANN_a81.mis, p_hat1500-3.mis	10
MWCP	BHSOLIB	50-23-5.clq, 53-24-4.clq, 56-25-3.clq, 59-26-1.clq	30
MWCP	DIMACS	brock800_4.clq, C4000.5.clq, DSJC1000.5.clq, gen400_p0.9_75.clq, hamming10-4.clq, keller6.clq, MANN_a81.clq, p_hat1500-3.clq	10

Table 3: The detail about the training set, where 't.t.' means the evaluation time of each instance during training.

Training

- average scores of the training set as the evaluate metric
- 24 hours for the entire configuration process

Content

Setting

Benchmarks

- MWDS: T1[7], T2[7], UDG[7], BHOSLIB[12], DIMACS[6]
- MWCP: BHOSLIB[12], DIMACS[6]

Setting

Benchmarks

- MWDS: T1[7], T2[7], UDG[7], BHOSLIB[12], DIMACS[6]
- MWCP: BHOSLIB[12], DIMACS[6]

Competitors

- MWDS
  - CC^2FS[11]
  - CC^2FS-P (ours, one PC for each COP)
  - CC^2FS-SP (ours, one PC for each benchmark)
- MWCP
  - LSCC[10]
  - LSCC-P (ours, one PC for each COP)
  - LSCC-SP(ours, one PC for each benchmark)

Comparison PCC with CC in MWDS

Table 4: Comparison results of CC^{2}FS and CC^{2}FS-P on 7 benchmarks. We also report the sum of difference ('sum of diff') between CC^{2}FS-P and CC^{2}FS in 'min' and 'avg'. The smaller the difference, the better for CC^{2}FS-P.

	Benchmarks	#inst.	CC^{2}FS		CC^{2}FS-P		sum of diff	
			#w.m.	#w.a.	#w.m.	#w.a.	min	avg
train	T1	5	0	0	**5**	**5**	**-167**	**-167.0**
train	T2	5	0	0	**3**	**3**	**-49**	**-49.0**
train	UDG	4	**4**	**4**	0	0	30	30.0
train	BHOSLIB	4	0	0	**1**	**4**	**-1**	**-8.4**
train	DIMACS	8	0	0	0	0	0	0.0
test	T1	525	10	10	**152**	**152**	**-2129**	**-2129.0**
test	T2	525	0	0	**57**	**57**	**-697**	**-697.0**
test	UDG	116	**81**	**81**	16	16	541	541.0
test	BHOSLIB	37	0	0	**6**	**26**	**-52**	**-105.1**
test	DIMACS	29	0	0	**2**	**9**	**-4**	**-27.2**
Total		1258	95	95	**242**	**272**	**-2528**	**-2611.7**

- CC^{2}FS-P is the better solver (except the UDG)
- CC^{2}FS-P performs worse even in the training set for UDG

Comparison PCC with CC in MWDS

Table 4: Comparison results of CC^{2}FS and CC^{2}FS-P on 7 benchmarks. We also report the sum of difference ('sum of diff') between CC^{2}FS-P and CC^{2}FS in 'min' and 'avg'. The smaller the difference, the better for CC^{2}FS-P.

	Benchmarks	#inst.	CC^{2}FS #w.m.	CC^{2}FS #w.a.	CC^{2}FS-P #w.m.	CC^{2}FS-P #w.a.	sum of diff min	sum of diff avg
train	T1	5	0	0	**5**	**5**	**-167**	**-167.0**
train	T2	5	0	0	**3**	**3**	**-49**	**-49.0**
train	UDG	4	**4**	**4**	0	0	30	30.0
train	BHOSLIB	4	0	0	**1**	**4**	**-1**	**-8.4**
train	DIMACS	8	0	0	0	0	0	0.0
test	T1	525	10	10	**152**	**152**	**-2129**	**-2129.0**
test	T2	525	0	0	**57**	**57**	**-697**	**-697.0**
test	UDG	116	**81**	**81**	16	16	541	541.0
test	BHOSLIB	37	0	0	**6**	**26**	**-52**	**-105.1**
test	DIMACS	29	0	0	**2**	**9**	**-4**	**-27.2**
Total		1258	95	95	**242**	**272**	**-2528**	**-2611.7**

- CC^{2}FS-P is the better solver (except the UDG)
- CC^{2}FS-P performs worse even in the training set for UDG

the adverse bias generated from training data from other benchmarks

Comparison PCC with CC in MWCP

Table 5: Experimental results of LSCC and LSCC-P on DIMACS and BHOSLIB benchmarks. We report the sum of difference between LSCC-P and LSCC in 'min' and 'avg'. The larger the difference, the better for LSCC-P.

	Benchmarks	#inst.	LSCC #w.m.	LSCC #w.a.	LSCC-P #w.m.	LSCC-P #w.a.	sum of diff min	sum of diff avg
train	BHOSLIB	4	1	0	**2**	**4**	**13**	**67**
train	DIMACS	8	0	1	**3**	**3**	**124**	**166.8**
test	BHOSLIB	37	2	3	**14**	**26**	**315**	**462.7**
test	DIMACS	29	**2**	2	0	2	-19	**52.9**
Total		78	5	6	**19**	**35**	**433**	**749.4**

- LSCC-P achieves better performance than LSCC in BHOSLIB
- LSCC-P achieves comparable performance with LSCC in DIMACS

Discussion About the Bias of Training in MWDS

Table 6: Comparison results of CC^{2}FS and CC^{2}FS-SP.

	Benchmarks	#inst.	CC^{2}FS #w.m.	CC^{2}FS #w.a.	CC^{2}FS-SP #w.m.	CC^{2}FS-SP #w.a.	sum of diff min	sum of diff avg
train	T1	5	0	0	**5**	**5**	**-184**	**-184.0**
train	T2	5	0	0	**3**	**3**	**-49**	**-49.0**
train	UDG	4	1	1	**2**	**2**	**-8**	**-8.0**
train	BHOSLIB	4	0	0	**1**	**4**	**-1**	**-8.4**
train	DIMACS	8	0	0	0	0	0	0.0
test	T1	525	10	10	**154**	**154**	**-2205**	**-2205.0**
test	T2	525	0	0	**58**	**58**	**-697**	**-697.0**
test	UDG	116	**47**	**47**	41	41	**-14**	**-14.0**
test	BHOSLIB	37	0	0	**6**	**26**	**-52**	**-109.1**
test	DIMACS	29	0	0	**2**	**9**	**-4**	**-27.1**
Total		1258	58	58	**272**	**302**	**-3214**	**-3301.6**

Table 7: Comparison results of CC^{2}FS-P and CC^{2}FS-SP.

	Benchmarks	#inst.	CC^{2}FS-P #w.m.	CC^{2}FS-P #w.a.	CC^{2}FS-SP #w.m.	CC^{2}FS-SP #w.a.	sum of diff min	sum of diff avg
train	T1	5	0	0	**2**	**2**	**-17**	**-17.0**
train	T2	5	0	0	0	0	0	0.0
train	UDG	4	0	0	**4**	**4**	**-38**	**-38.0**
train	BHOSLIB	4	0	0	0	0	0	0.0
train	DIMACS	8	0	0	0	0	0	0.0
test	T1	525	17	17	**36**	**36**	**-76**	**-76.0**
test	T2	525	1	1	1	1	0	0.0
test	UDG	116	22	22	**82**	**82**	**-555**	**-555.0**
test	BHOSLIB	37	0	0	0	**1**	0	**-4.0**
test	DIMACS	29	0	**1**	0	0	0	0.1
Total		1258	40	41	**125**	**126**	**-686**	**-689.9**

- CC^{2}FS-SP achieves a better performance than CC^{2}FS-P
- CC^{2}FS-SP and CC^{2}FS are comparable in UDG

Discussion About the Bias of Training in MWCP

Table 8: Comparison results of LSCC and LSCC-SP.

	Benchmarks	#inst.	LSCC #w.m.	LSCC #w.a.	LSCC-SP #w.m.	LSCC-SP #w.a.	sum of diff max	sum of diff avg
train	BHOSLIB	4	1	0	**2**	**4**	**8**	**69.4**
train	DIMACS	8	0	1	**3**	**3**	**124**	**166.8**
test	BHOSLIB	37	2	2	**14**	**27**	**371**	**502.3**
test	DIMACS	29	**2**	2	0	2	-19	**52.9**
Total		78	5	5	**19**	**36**	**484**	**791.4**

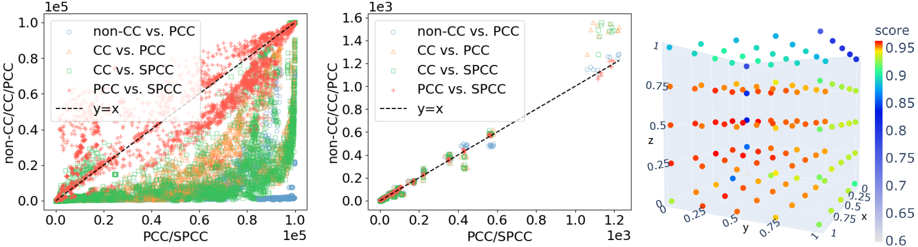
Table 9: Comparison results of LSCC-P and LSCC-SP.

	Benchmarks	#inst.	LSCC-P #w.m.	LSCC-P #w.a.	LSCC-SP #w.m.	LSCC-SP #w.a.	sum of diff max	sum of diff avg
train	BHOSLIB	4	1	2	**2**	2	-5	**2.4**
train	DIMACS	8	0	0	0	0	0	0
test	BHOSLIB	37	**9**	8	6	**15**	**56**	**39.6**
test	DIMACS	29	0	0	0	0	0	0
Total		78	**10**	10	8	**17**	**51**	**42**

- specially PC for each benchmark can improve the performance
- in DIMACS, LSCC-SP and LSCC-P are the same because they share the same PC

Performance on Reducing the Cycling Problem

Figure 5: (a) Comparison of the number of different candidate solutions of non-CC, CC, PCC, SPCC. (b) Comparison of CPU time. (c) The scores of CC^{2}FS-P with different configurations on MWDS benchmark. The X-axis, Y-axis, and Z-axis represent p_1, p_2, and p_3 respectively.



(a) (b) (c)

- not have much more time consumption
- potentially alleviates the cycling problem
- necessary to tune for an optimal PC

Content

Conclusion and Future Work

Conclusion:

1. We observe that neighborhoods with different levels should have different contributions to alleviate the cycling problem.
2. We expand the configuration with probability, i.e., probabilistic configuration (PC), to capture the contributions of vertices at different levels, resulting in the probabilistic configuration checking (PCC) strategy.
3. Our experimental results confirm that the PC can improve existing local search algorithms in two classic COPs, i.e., MWDS and MWCP, due to alleviating the cycling problem.

Conclusion and Future Work

Conclusion:

1. We observe that neighborhoods with different levels should have different contributions to alleviate the cycling problem.
2. We expand the configuration with probability, i.e., probabilistic configuration (PC), to capture the contributions of vertices at different levels, resulting in the probabilistic configuration checking (PCC) strategy.
3. Our experimental results confirm that the PC can improve existing local search algorithms in two classic COPs, i.e., MWDS and MWCP, due to alleviating the cycling problem.

Future work:

1. making the method suitable for massive graphs.
2. extending our approach to other COPs.

References I

[1] Cai, S.; and Su, K. 2013. Local search for Boolean Satisfiability with configuration checking and subscore. *Artificial Intelligence*, 204: 75–98.

[2] Cai, S.; Su, K.; Luo, C.; and Sattar, A. 2013. NuMVC: An efficient local search algorithm for minimum vertex cover. *Journal of Artificial Intelligence Research*, 46: 687–716.

[3] Cai, S.; Su, K.; and Sattar, A. 2011. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. *Artificial Intelligence*, 175(9-10): 1672–1696.

[4] Chen, P.; Wan, H.; Cai, S.; Li, J.; and Chen, H. 2020. Local Search with Dynamic-Threshold Configuration Checking and Incremental Neighborhood Updating for Maximum k-plex Problem. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, (AAAI 2020)*, 2343–2350.

[5] Hutter, F.; Hoos, H. H.; and Leyton-Brown, K. 2011. Sequential model-based optimization for general algorithm configuration. In *LION*, 507–523.

[6] Johnson, D. S.; and Trick, M. A. 1996. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc.

[7] Jovanovic, R.; Tuba, M.; and Simian, D. 2010. Ant colony optimization applied to minimum weight dominating set problem. In *Proceedings of the 12th WSEAS international conference on Automatic control, modelling & simulation, (ACMOS-2010)*, 322–326.

[8] Wang, Y.; Cai, S.; Chen, J.; and Yin, M. 2018. A Fast Local Search Algorithm for Minimum Weight Dominating Set Problem on Massive Graphs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, (IJCAI-2018)*, 1514–1522.

[9] Wang, Y.; Cai, S.; Chen, J.; and Yin, M. 2020. SCCWalk: An efficient local search algorithm and its improvements for maximum weight clique problem. *Journal of Artificial Intelligence*, 280: 103230–103263.

References II

[10] Wang, Y.; Cai, S.; and Yin, M. 2016. Two efficient local search algorithms for maximum weight clique problem. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, (AAAI-2016)*, 805–811.

[11] Wang, Y.; Cai, S.; and Yin, M. 2017. Local Search for Minimum Weight Dominating Set with Two-Level Configuration Checking and Frequency Based Scoring Function. *Journal of Artificial Intelligence Research*, 58: 267–295. ArXiv: 1702.04594.

[12] Wu, Q.; Hao, J.-K.; and Glover, F. 2012. Multi-neighborhood tabu search for the maximum weight clique problem. *Annals of Operations Research*, 196(1): 611–634.

Thank you for your listening!