On the solution of some very large-scale, highly degenerate combinatorial optimization problems: Applications to clustering

D. Aloise **C. Contardo**

ESG UQAM, GERAD and CIRRELT

Column Generation 2016, Búzios, Brazil, 2016

Outline

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

- $U \subseteq V \implies f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

- $U \subseteq V \implies f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

- $U \subseteq V \Longrightarrow f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

With the three additional properties:

- $U \subseteq V \implies f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

With the three additional properties:

- $U \subseteq V \Longrightarrow f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

With the three additional properties:

- $U \subseteq V \Longrightarrow f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

With the three additional properties:

- $U \subseteq V \Longrightarrow f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V, |U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

What problems are we interested in?

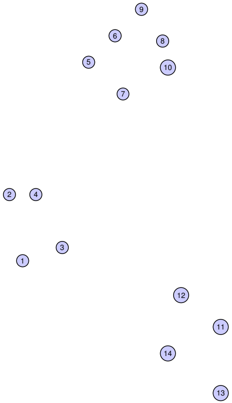
- Given a set V,
- find a *partition* $P(V)$ of the set V
- so as to minimize a cost function $f(P(V))$

With the three additional properties:

- $U \subseteq V \implies f(P^*(U)) \leq f(P^*(V))$ (monotonicity)
- $\exists U \subseteq V$, $|U| \ll |V|$ such that
  - $f(P^*(U)) = f(P^*(V))$ (degeneracy)
  - Possible to build $P^*(V)$ from enlarging $P^*(U)$ (constructibility)

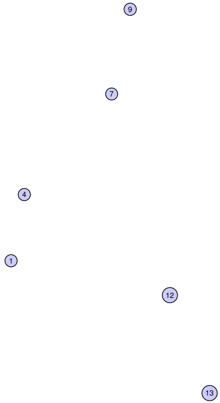
The main theorem

- Let us consider a set V of nodes
- Let us consider a subset $U \subseteq V$ of observations
- Let $P^*(U), f^*(U)$ be the optimal partition and its cost
- If a node $v \in V \setminus U$ can be added to $P^*(U)$ to form P' and $f(P') = f^*(U)$ then $P' = P^*(U \cup \{v\})$ and $f(P') = f^*(U \cup \{v\})$

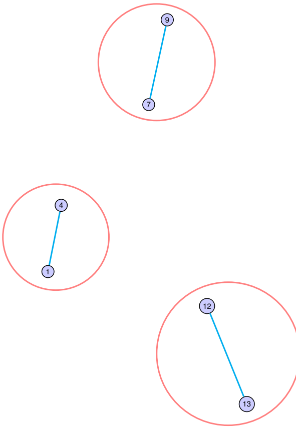


The main theorem

- Let us consider a set V of nodes
- Let us consider a subset $U \subseteq V$ of observations
- Let $P^*(U), f^*(U)$ be the optimal partition and its cost
- If a node $v \in V \setminus U$ can be added to $P^*(U)$ to form P' and $f(P') = f^*(U)$ then $P' = P^*(U \cup \{v\})$ and $f(P') = f^*(U \cup \{v\})$



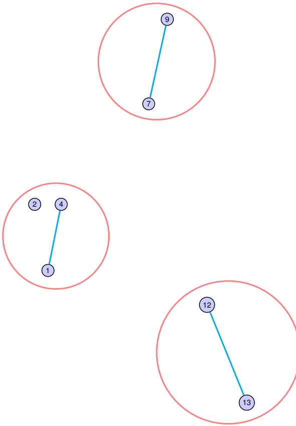
The main theorem



- Let us consider a set V of nodes
- Let us consider a subset $U \subseteq V$ of observations
- Let $P^*(U)$, $f^*(U)$ be the optimal partition and its cost
- If a node $v \in V \setminus U$ can be added to $P^*(U)$ to form P' and $f(P') = f^*(U)$ then $P' = P^*(U \cup \{v\})$ and $f(P') = f^*(U \cup \{v\})$

The main theorem

- Let us consider a set V of nodes
- Let us consider a subset $U \subseteq V$ of observations
- Let $P^*(U)$, $f^*(U)$ be the optimal partition and its cost
- If a node $v \in V \setminus U$ can be added to $P^*(U)$ to form P' and $f(P') = f^*(U)$ then $P' = P^*(U \cup \{v\})$ and $f(P') = f^*(U \cup \{v\})$



The algorithm

Algorithm 1 Chunking method

Require: Set V, function f, number of clusters k

Ensure: Optimal partition $P(V)$ that minimizes $f(P(V))$

$U \leftarrow \emptyset, f^U, f^V \leftarrow \infty, P^U, P^V \leftarrow \emptyset, W \leftarrow \emptyset$

repeat

$U \leftarrow U \cup W$

$(f^U, P^U) \leftarrow \texttt{ExactSPP}(U, k)$

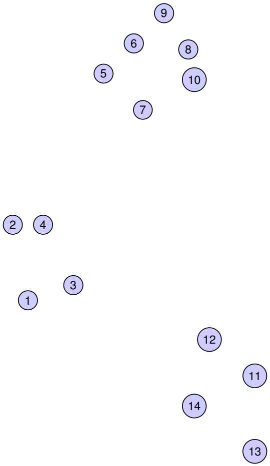
$(f^V, P^V, W) \leftarrow \texttt{HeuristicSPP}(P^U, V \setminus U)$

until $W = \emptyset$

return P^V

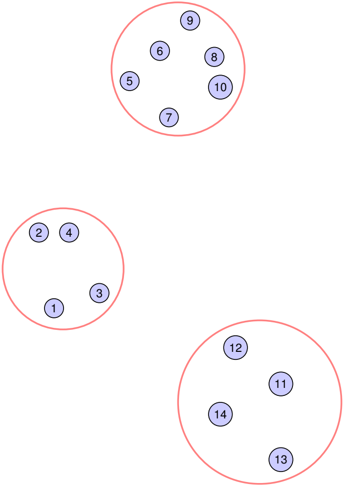
Clustering problems

- We are given a set V of n observations
- Observations must be partitioned
- A cluster must contain similar observations



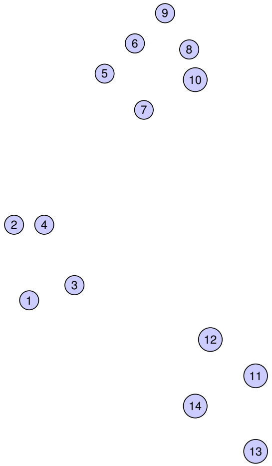
Clustering problems

- We are given a set V of n observations
- Observations must be partitioned
- A cluster must contain similar observations



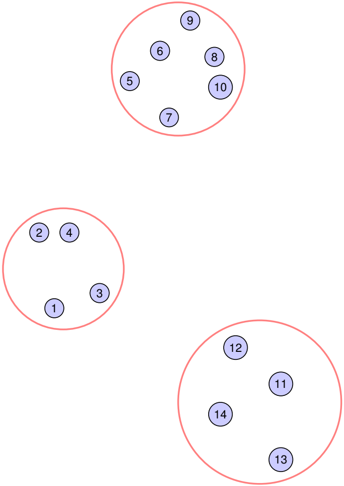
The minimax diameter clustering problem

- Strongly NP-hard (Garey & Johnson 1979)
- Objective: Minimize the maximum intra-cluser dissimilarity

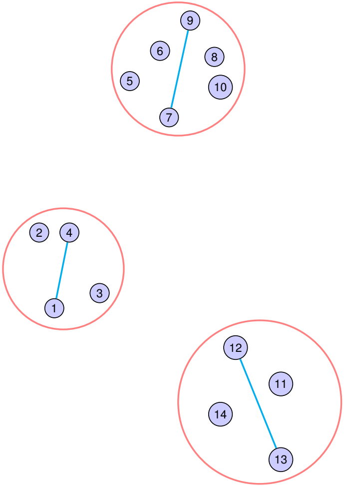


The minimax diameter clustering problem

- Strongly NP-hard (Garey & Johnson 1979)
- Objective: Minimize the maximum intra-cluser dissimilarity



The minimax diameter clustering problem

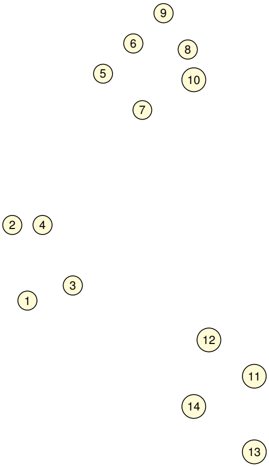


- Strongly NP-hard (Garey & Johnson 1979)
- Objective: Minimize the maximum intra-cluser dissimilarity

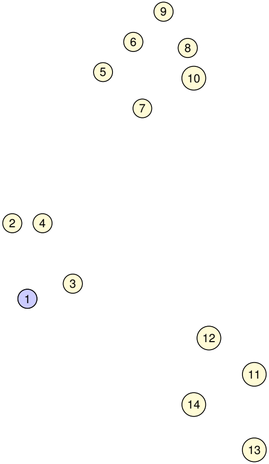
State-of-the-art algorithms

- Complete linkage: most popular heuristic method
- Constraint Programming: most efficient exact method
  - Can solve problems containing up to 5k observations
- Both methods:
  - need to compute and store the dissimilarity matrix (cpu = mem = $O(n^2)$)
  - therefore, they cannot be applied to large problems

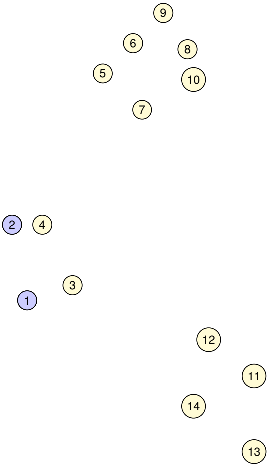
Chunking algorithm



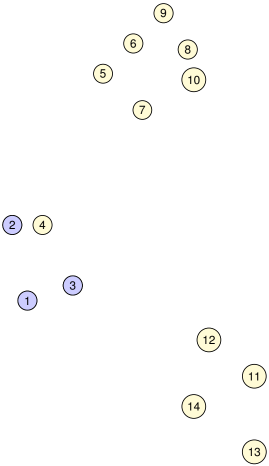
Chunking algorithm



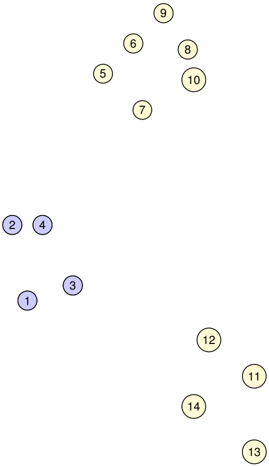
Chunking algorithm



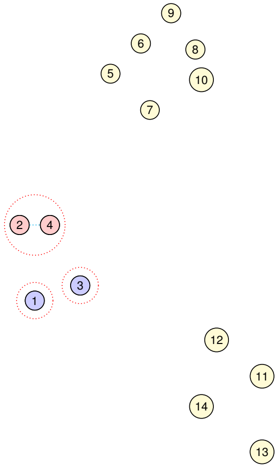
Chunking algorithm



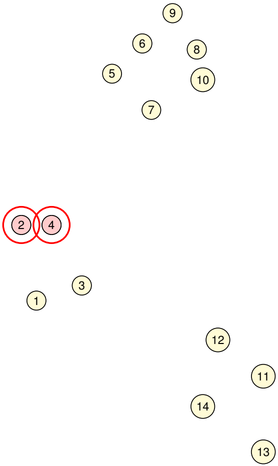
Chunking algorithm



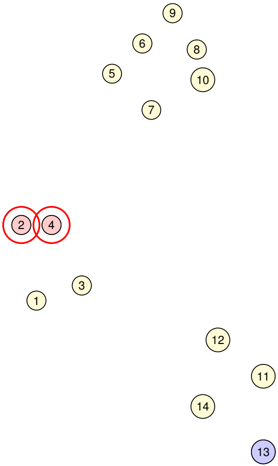
Chunking algorithm



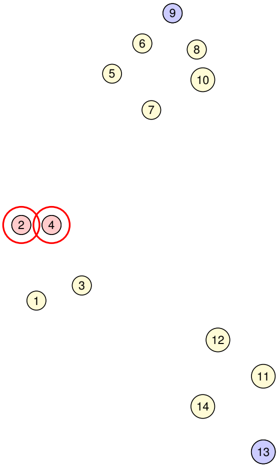
Chunking algorithm



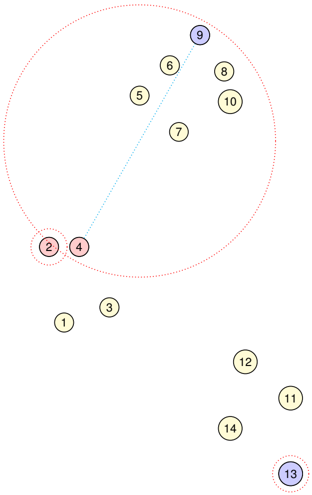
Chunking algorithm



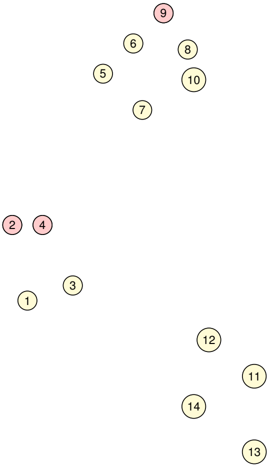
Chunking algorithm



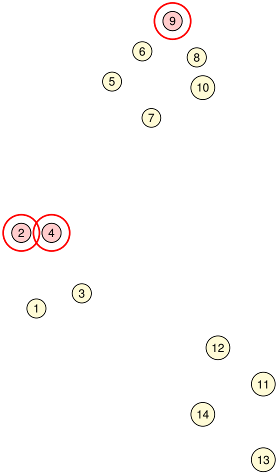
Chunking algorithm



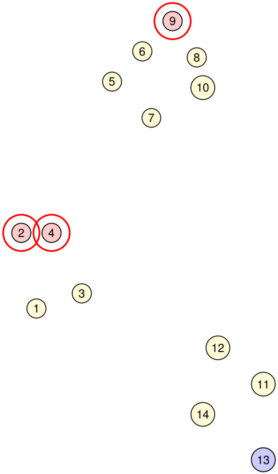
Chunking algorithm



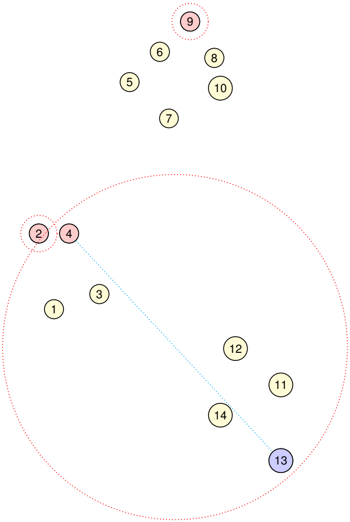
Chunking algorithm



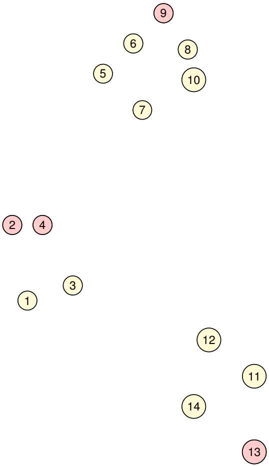
Chunking algorithm



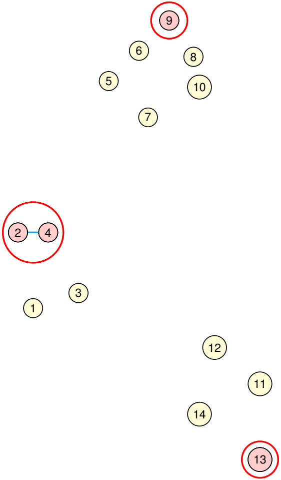
Chunking algorithm



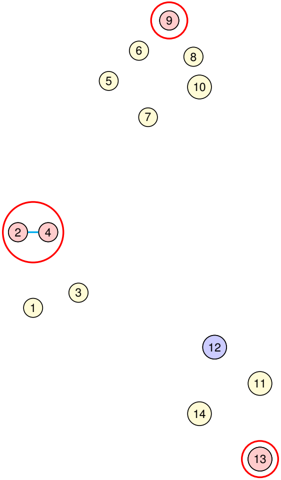
Chunking algorithm



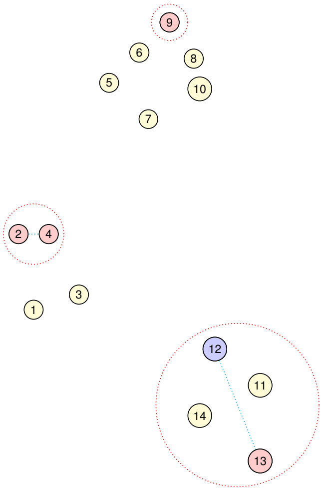
Chunking algorithm



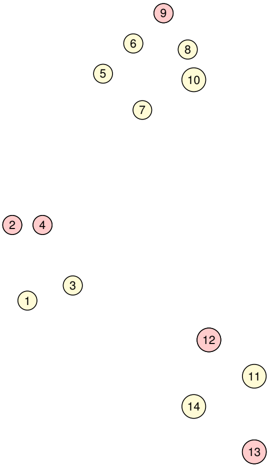
Chunking algorithm



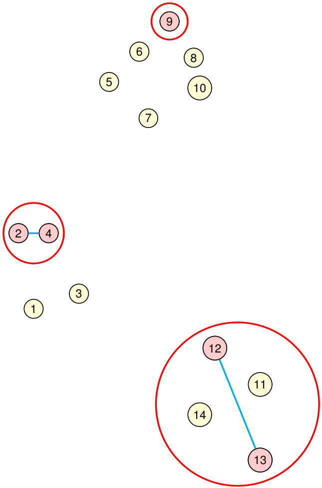
Chunking algorithm



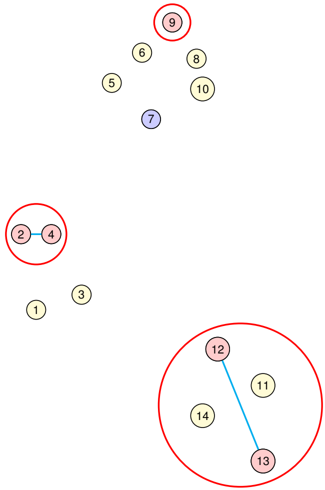
Chunking algorithm



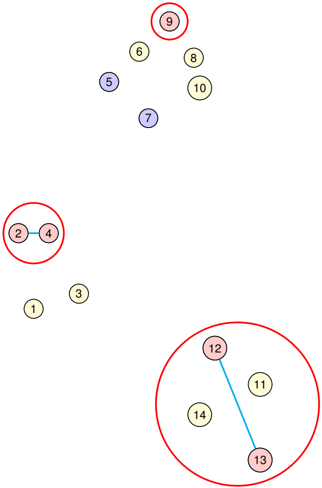
Chunking algorithm



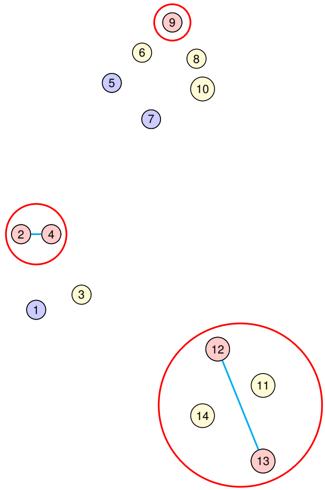
Chunking algorithm



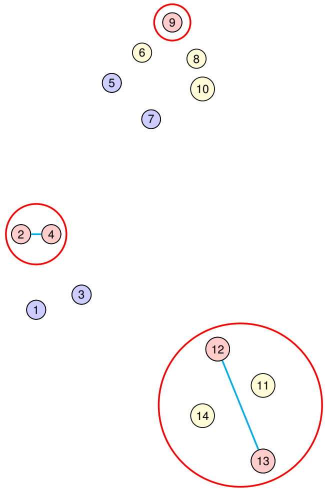
Chunking algorithm



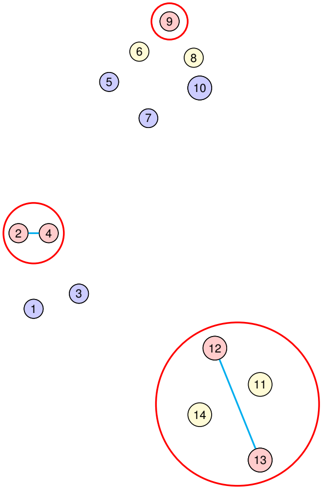
Chunking algorithm



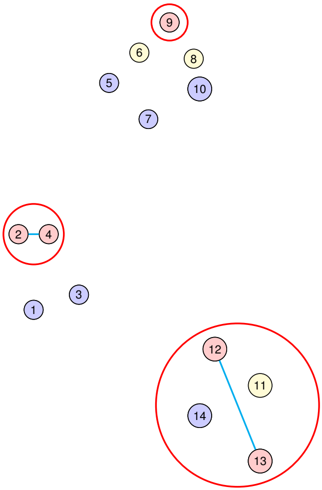
Chunking algorithm



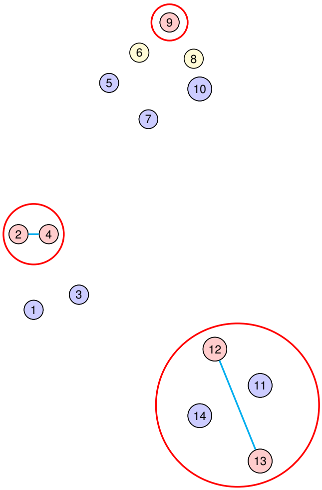
Chunking algorithm



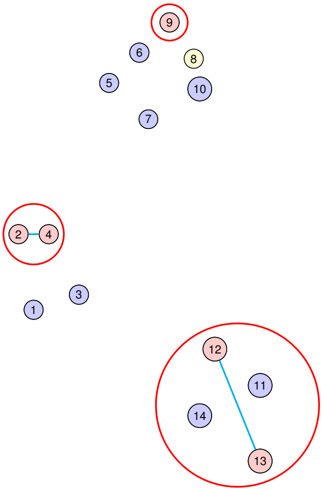
Chunking algorithm



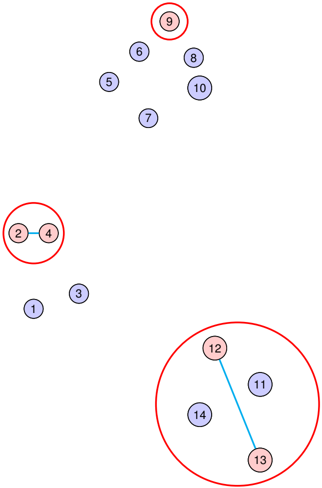
Chunking algorithm



Chunking algorithm



Chunking algorithm



Observations

- Complete problem contains 14 observations
- Largest `ExactSPP(U, k)` contains only 5 nodes
- The dissimilarity matrix must only be stored for these smaller problems
  - No storage problems
  - In practice, our method is faster than computing the dissimilarity matrix (still $O(n^2)$ in practice though)
- Ordering of the nodes for the heuristic is critical (most likely to result in an infeasible insertion are inspected first)
- The bottleneck of our algorithm is `HeuristicSPP`$(P^U, V \setminus U)$! ($O(n^3)$ in the worst case but $O(n^2)$ in practice)

Observations

- Complete problem contains 14 observations
- Largest `ExactSPP(U, k)` contains only 5 nodes
- The dissimilarity matrix must only be stored for these smaller problems
  - No storage problems
  - In practice, our method is faster than computing the dissimilarity matrix (still $O(n^2)$ in practice though)
- Ordering of the nodes for the heuristic is critical (most likely to result in an infeasible insertion are inspected first)
- The bottleneck of our algorithm is `HeuristicSPP(`$P^U, V \setminus U$`)` ! ($O(n^3)$ in the worst case but $O(n^2)$ in practice)

Observations

- Complete problem contains 14 observations
- Largest `ExactSPP(U, k)` contains only 5 nodes
- The dissimilarity matrix must only be stored for these smaller problems
  - No storage problems
  - In practice, our method is faster than computing the dissimilarity matrix (still $O(n^2)$ in practice though)
- Ordering of the nodes for the heuristic is critical (most likely to result in an infeasible insertion are inspected first)
- The bottleneck of our algorithm is `HeuristicSPP`($P^U, V \setminus U$)! ($O(n^3)$ in the worst case but $O(n^2)$ in practice)

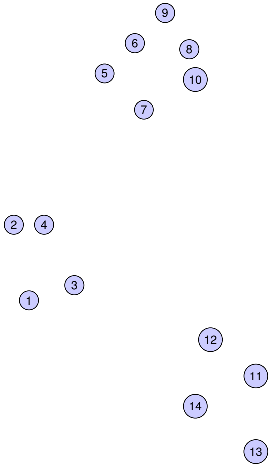
Observations

- Complete problem contains 14 observations
- Largest `ExactSPP(U, k)` contains only 5 nodes
- The dissimilarity matrix must only be stored for these smaller problems
  - No storage problems
  - In practice, our method is faster than computing the dissimilarity matrix (still $O(n^2)$ in practice though)
- Ordering of the nodes for the heuristic is critical (most likely to result in an infeasible insertion are inspected first)
- The bottleneck of our algorithm is `HeuristicSPP`($P^U, V \setminus U$)! ($O(n^3)$ in the worst case but $O(n^2)$ in practice)

Observations

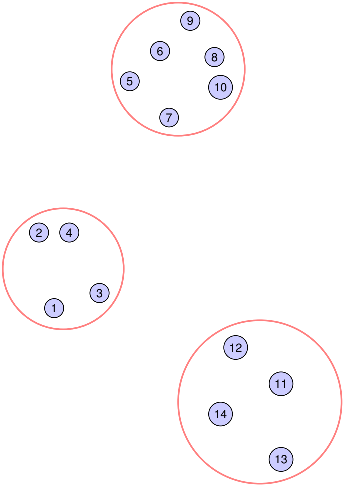
- Complete problem contains 14 observations
- Largest `ExactSPP(U, k)` contains only 5 nodes
- The dissimilarity matrix must only be stored for these smaller problems
  - No storage problems
  - In practice, our method is faster than computing the dissimilarity matrix (still $O(n^2)$ in practice though)
- Ordering of the nodes for the heuristic is critical (most likely to result in an infeasible insertion are inspected first)
- The bottleneck of our algorithm is `HeuristicSPP(`$P^U, V \setminus U$`)`! ($O(n^3)$ in the worst case but $O(n^2)$ in practice)

Maximum split clustering problem



- Polynomially solvable
- cpu = mem = $O(n^2)$
- Objective: Maximize the minimum inter-cluser dissimilarity

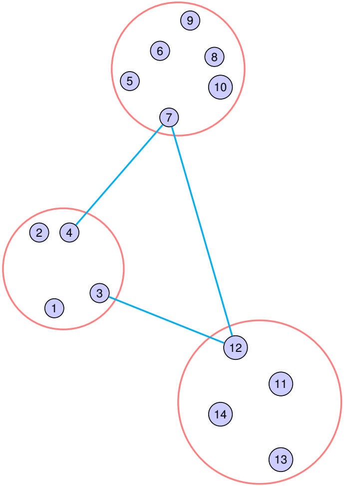
Maximum split clustering problem



- Polynomially solvable
- cpu = mem = $O(n^2)$
- Objective: Maximize the minimum inter-cluser dissimilarity

Maximum split clustering problem

- Polynomially solvable
- cpu = mem = $O(n^2)$
- Objective: Maximize the minimum inter-cluser dissimilarity



Other higly degenerate clustering criteria

- Maximize the ratio $\left(\frac{Split}{Diameter}\right)$
- Minimize a convex combination of *Diameter* and $-Split$
- Minimize the (weighted) sum of the diameters
- Maximize the (weighted) sum of the splits
- Combinations of the above

Computational experience

Problem	n	m	k	d
Iris	150	11,175	3	4
Wine	178	15,753	3	13
Glass	214	22,791	7	9
Ionosphere	351	61,425	2	34
User knowledge	403	81,003	4	5
Breast cancer	569	161,596	2	30
Synthetic control	600	179,700	6	60
Vehicle	846	357,435	4	18
Yeast	1,484	1,100,386	10	8
Mfeat (morph)	2,000	1,999,000	10	6
Multiple features	2,000	1,999,000	10	649
Segmentation	2,000	1,999,000	7	19
Image segm	2,310	2,666,895	7	19
Waveform (v1)	5,000	12,497,500	3	21
Waveform (v2)	5,000	12,497,500	3	40
Ailerons	13,750	94,524,375	10	41
Magic	19,020	180,870,690	2	10
Krkopt	28,056	393,555,540	17	6
Shuttle	58,000	1,681,971,000	7	9
Connect-4	67,557	2,281,940,346	3	42
SensIt (acoustic)	96,080	4,615,635,160	3	50
Twitter	140,707	9,899,159,571	2	77
Census	142,521	10,156,046,460	3	41
HAR	165,633	13,717,062,528	5	18
IJCNN1	191,681	18,370,707,040	2	22
Cod-Rna	488,565	119,347,635,330	2	8
KDD cup 10%	494,090	122,062,217,005	23	41
Cover type	581,012	168,787,181,566	7	54

Table: Problems details

Computational experience

	Problem	n	m	k	d
State-of-the-art	Iris	150	11,175	3	4
	Wine	178	15,753	3	13
	Glass	214	22,791	7	9
	Ionosphere	351	61,425	2	34
	User knowledge	403	81,003	4	5
	Breast cancer	569	161,596	2	30
	Synthetic control	600	179,700	6	60
	Vehicle	846	357,435	4	18
	Yeast	1,484	1,100,386	10	8
	Mfeat (morph)	2,000	1,999,000	10	6
	Multiple features	2,000	1,999,000	10	649
	Segmentation	2,000	1,999,000	7	19
	Image segm	2,310	2,666,895	7	19
	Waveform (v1)	5,000	12,497,500	3	21
	Waveform (v2)	5,000	12,497,500	3	40
	Ailerons	13,750	94,524,375	10	41
	Magic	19,020	180,870,690	2	10
	Krkopt	28,056	393,555,540	17	6
	Shuttle	58,000	1,681,971,000	7	9
	Connect-4	67,557	2,281,940,346	3	42
	SensIt (acoustic)	96,080	4,615,635,160	3	50
	Twitter	140,707	9,899,159,571	2	77
	Census	142,521	10,156,046,460	3	41
	HAR	165,633	13,717,062,528	5	18
	IJCNN1	191,681	18,370,707,040	2	22
	Cod-Rna	488,565	119,347,635,330	2	8
	KDD cup 10%	494,090	122,062,217,005	23	41
	Cover type	581,012	168,787,181,566	7	54

Table: Problems details

Computational experience

	Problem	n	m	k	d
State-of-the-art	Iris	150	11,175	3	4
	Wine	178	15,753	3	13
	Glass	214	22,791	7	9
	Ionosphere	351	61,425	2	34
	User knowledge	403	81,003	4	5
	Breast cancer	569	161,596	2	30
	Synthetic control	600	179,700	6	60
	Vehicle	846	357,435	4	18
	Yeast	1,484	1,100,386	10	8
	Mfeat (morph)	2,000	1,999,000	10	6
	Multiple features	2,000	1,999,000	10	649
	Segmentation	2,000	1,999,000	7	19
	Image segm	2,310	2,666,895	7	19
	Waveform (v1)	5,000	12,497,500	3	21
	Waveform (v2)	5,000	12,497,500	3	40
Too large to fit in ram	Ailerons	13,750	94,524,375	10	41
	Magic	19,020	180,870,690	2	10
	Krkopt	28,056	393,555,540	17	6
	Shuttle	58,000	1,681,971,000	7	9
	Connect-4	67,557	2,281,940,346	3	42
	SensIt (acoustic)	96,080	4,615,635,160	3	50
	Twitter	140,707	9,899,159,571	2	77
	Census	142,521	10,156,046,460	3	41
	HAR	165,633	13,717,062,528	5	18
	IJCNN1	191,681	18,370,707,040	2	22
	Cod-Rna	488,565	119,347,635,330	2	8
	KDD cup 10%	494,090	122,062,217,005	23	41
	Cover type	581,012	168,787,181,566	7	54

Table: Problems details

Computational experience

Problem	Opt	RBBA	BB	CP	IC
Iris	2.58	1.4	1.8	< 0.1	< 0.1
Wine	458.13	2.0	2.3	< 0.1	< 0.1
Glass	4.97	8.1	42.0	0.2	0.2
Ionosphere	8.6		0.6	0.3	0.2
User knowledge	1.17		3.7	0.2	1.2
Breast cancer	2,377.96		1.8	0.5	0.2
Synthetic control	109.36			1.6	0.4
Vehicle	264.83			0.9	0.2
Yeast	0.67			5.2	1.7
Mfeat (morph)	1,594.96			8.59	0.6
Segmentation	436.4			5.7	0.6
Waveform (v2)	15.58			50.1	2.0

Table: Running times (in seconds) on small datasets

Computational experience

Problem	Opt	Chunking method				dmc
		it	n'	lch	t	
Waveform (v1)	13.74	10	21	< 0.1	< 0.1	< 0.1
Waveform (v2)	15.58	9	22	< 0.1	< 0.1	< 0.1
Ailerons	230.71	34	49	< 0.1	0.2	0.17
Magic	692.44	3	12	0.33	0.37	0.27
Krkopt	2.00	60	77	< 0.1	0.39	0.47
Shuttle	6,157.44	5	14	3.23	3.38	2.95
Connect-4	3.87	11	20	2.31	2.73	6.45
SensIt (acoustic)	4.47	6	15	12.72	13.14	12.16
Twitter	80,734	2	11	28.19	28.77	27.91
Census	100,056	3	13	33.27	33.95	33.00
HAR	1,078.73	8	18	18.70	19.25	24.76
IJCNN1	3.97	5	14	12.98	13.36	17.90
Cod-Rna	934.68	3	12	122.86	123.62	97.26
KDD cup 10%	144,165	26	53	25.50	28.43	23.71
Cover type	3,557.3	129	143	122.5	162.94	393.35

Table: Detailed results on the chunking method

Conclusions

- Our method seems to be very sensitive to noise. Noisy problems (Birch1, 2, 3, pendigits) could not be solved although smaller in size
- We are currently testing our framework to solve other classification problems presenting high degrees of degeneracy
- The method seems easily adaptable to become a heuristic capable of handling larger problems (as in *online streaming of data*)