Compacting Points-To Sets through Object Clustering

Mohamad Barbar [1,2]   Yulei Sui [1]

[1]University of Technology Sydney, Australia

[2]CSIRO's Data61, Australia

OOPSLA '21

Points-To Analysis

Points-To Analysis

Determine what each pointer points to.

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, \ldots\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, \ldots\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

- Flow

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, \ldots\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

- Flow
- Context

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

- Flow
- Context
- Field

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

- Flow
- Context
- Field
- ...

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, ...\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

- Flow
- Context
- Field
- ...

(generally)
Precision $\Rightarrow$ higher cost

Points-To Analysis

Determine what each pointer points to.

$$pt(p) = \{o_1, o_2, o_3, \ldots\}$$

Bit-vector representations have been successfully and widely used in mainstream points-to analysis frameworks.

Why?

- Bug detection
- Optimisation
- Instrumentation

Various sensitivities

- *Flow*
- Context
- Field
- ...

(generally)
Precision $\Rightarrow$ higher cost

Staged Flow-Sensitive Points-to Analysis (SFS)

Staged Flow-Sensitive Points-to Analysis (SFS)

1. Perform fast auxiliary analysis (e.g. flow-insensitive Andersen's analysis).

Staged Flow-Sensitive Points-to Analysis (SFS)

1. Perform fast auxiliary analysis (e.g. flow-insensitive Andersen's analysis).
2. Build over-approximate memory SSA form.

Staged Flow-Sensitive Points-to Analysis (SFS)

1. Perform fast auxiliary analysis (e.g. flow-insensitive Andersen's analysis).
2. Build over-approximate memory SSA form.
3. Build def-use graph.

Staged Flow-Sensitive Points-to Analysis (SFS)

1. Perform fast auxiliary analysis (e.g. flow-insensitive Andersen's analysis).
2. Build over-approximate memory SSA form.
3. Build def-use graph.
4. Perform flow-sensitive analysis on def-use graph, not control-flow graph.

Motivation

Why do compact points-to set representations matter?

Program	Program Points	Top-Level Variables	Memory Objects	Average PTS	Largest PTS	SFS Unions
dhcpcd	57 168	63 196	3701	223.55	265	90 785 902
gawk	279 931	141 136	4784	434.81	811	2 275 388 148
bash	254 314	149 070	4339	244.57	324	531 039 266
mutt	360 535	178 147	7169	379.47	1238	1 347 064 278
lynx	579 285	237 252	7917	198.02	1129	4 829 162 478
xpdf	440 418	388 859	19 101	87.78	1590	12 423 325 697
ruby	795 643	670 649	20 235	2.68	1726	13 502 095 022
keepassxc	572 001	604 363	37 671	171.70	365	879 430 055

Motivation

Why do compact points-to set representations matter?

Program	Program Points	Top-Level Variables	Memory Objects	Average PTS	Largest PTS	SFS Unions
dhcpcd	57 168	63 196	3701	223.55	265	90 785 902
gawk	279 931	141 136	4784	434.81	811	2 275 388 148
bash	254 314	149 070	4339	244.57	324	531 039 266
mutt	360 535	178 147	7169	379.47	1238	1 347 064 278
lynx	579 285	237 252	7917	198.02	1129	4 829 162 478
xpdf	440 418	388 859	19 101	87.78	1590	12 423 325 697
ruby	795 643	670 649	20 235	2.68	1726	13 502 095 022
keepassxc	572 001	604 363	37 671	171.70	365	879 430 055

Motivation

Why do compact points-to set representations matter?

Program	Program Points	Top-Level Variables	Memory Objects	Average PTS	Largest PTS	SFS Unions
dhcpcd	57 168	63 196	3701	223.55	265	90 785 902
gawk	279 931	141 136	4784	434.81	811	2 275 388 148
bash	254 314	149 070	4339	244.57	324	531 039 266
mutt	360 535	178 147	7169	379.47	1238	1 347 064 278
lynx	579 285	237 252	7917	198.02	1129	4 829 162 478
xpdf	440 418	388 859	19 101	87.78	1590	12 423 325 697
ruby	795 643	670 649	20 235	2.68	1726	13 502 095 022
keepassxc	572 001	604 363	37 671	171.70	365	879 430 055

Motivation

Why do compact points-to set representations matter?

Program	Program Points	Top-Level Variables	Memory Objects	Average PTS	Largest PTS	SFS Unions
dhcpcd	57 168	63 196	3701	223.55	265	90 785 902
gawk	279 931	141 136	4784	434.81	811	2 275 388 148
bash	254 314	149 070	4339	244.57	324	531 039 266
mutt	360 535	178 147	7169	379.47	1238	1 347 064 278
lynx	579 285	237 252	7917	198.02	1129	4 829 162 478
xpdf	440 418	388 859	19 101	87.78	1590	12 423 325 697
ruby	795 643	670 649	20 235	2.68	1726	13 502 095 022
keepassxc	572 001	604 363	37 671	171.70	365	879 430 055

Motivation

Why do compact points-to set representations matter?

Program	Program Points	Top-Level Variables	Memory Objects	Average PTS	Largest PTS	SFS Unions
dhcpcd	57 168	63 196	3701	223.55	265	90 785 902
gawk	279 931	141 136	4784	434.81	811	2 275 388 148
bash	254 314	149 070	4339	244.57	324	531 039 266
mutt	360 535	178 147	7169	379.47	1238	1 347 064 278
lynx	579 285	237 252	7917	198.02	1129	4 829 162 478
xpdf	440 418	388 859	19 101	87.78	1590	12 423 325 697
ruby	795 643	670 649	20 235	2.68	1726	13 502 095 022
keepassxc	572 001	604 363	37 671	171.70	365	879 430 055

Motivation

Why do compact points-to set representations matter?

Program	Program Points	Top-Level Variables	Memory Objects	Average PTS	Largest PTS	SFS Unions
dhcpcd	57 168	63 196	3701	223.55	265	90 785 902
gawk	279 931	141 136	4784	434.81	811	2 275 388 148
bash	254 314	149 070	4339	244.57	324	531 039 266
mutt	360 535	178 147	7169	379.47	1238	1 347 064 278
lynx	579 285	237 252	7917	198.02	1129	4 829 162 478
xpdf	440 418	388 859	19 101	87.78	1590	12 423 325 697
ruby	795 643	670 649	20 235	2.68	1726	13 502 095 022
keepassxc	572 001	604 363	37 671	171.70	365	879 430 055

Bit-Vectors

String of bits implemented as an array of **words** to represent integral sets.

Bit-Vectors

String of bits implemented as an array of **words** to represent integral sets.

- Bit set? Index of the bit is a member of the set.

Bit-Vectors

String of bits implemented as an array of **words** to represent integral sets.

- Bit set? Index of the bit is a member of the set.
- Bit unset? Index of the bit is not a member of the set.

Bit-Vectors

String of bits implemented as an array of **words** to represent integral sets.

- Bit set? Index of the bit is a member of the set.
- Bit unset? Index of the bit is not a member of the set.

$$\{\ 0,\ 3,\qquad\qquad 8,9,\quad \}$$
$$[\,\langle 1001\rangle, \langle 0000\rangle, \langle 1100\rangle\,]$$

($\langle \times\times\times\times \rangle$ is a 4-bit word.)

Bit-Vectors: Fast Unions

$$
\begin{array}{lcccccc}
 & [& w_1, & w_2, & \dots, & w_n &] \\
| & [& w'_1, & w'_2, & \dots, & w'_n &] \\
\hline
 & [& w_1 | w'_1, & w_2 | w'_2, & \dots, & w_n | w'_n &].
\end{array}
$$

Bit-Vectors: Fast Unions

$$
\begin{array}{cccccccc}
 & [& w_1, & w_2, & \ldots, & w_n &] \\
| & [& w'_1, & w'_2, & \ldots, & w'_n &] \\
\hline
 & [& w_1|w'_1, & w_2|w'_2, & \ldots, & w_n|w'_n &].
\end{array}
$$

- Amenable to vectorisation.

Bit-Vectors: Fast Unions

$$
\begin{array}{cccccccc}
 & [& w_1, & w_2, & \ldots, & w_n &] \\
| & [& w'_1, & w'_2, & \ldots, & w'_n &] \\
\hline
 & [& w_1|w'_1, & w_2|w'_2, & \ldots, & w_n|w'_n &].
\end{array}
$$

- Amenable to vectorisation.
- Good spatial locality.

Bit-Vectors: Fast Unions

$$
\begin{array}{cccccccc}
 & [& w_1, & w_2, & \ldots, & w_n &] \\
| & [& w'_1, & w'_2, & \ldots, & w'_n &] \\
\hline
 & [& w_1|w'_1, & w_2|w'_2, & \ldots, & w_n|w'_n &].
\end{array}
$$

- Amenable to vectorisation.
- Good spatial locality.
- Compact; very little metadata.

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Consider the set $\{9995, 9996, 9997, 9998, 9999\}$.

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Consider the set $\{9995, 9996, 9997, 9998, 9999\}$.

Before stripping leading zeroes:

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Consider the set $\{9995, 9996, 9997, 9998, 9999\}$.

Before stripping leading zeroes:

$$[\,\underbrace{\langle 0000\rangle, \ldots, \langle 0000\rangle}_{\text{2498 words}}, \langle 000\,1_{9995}\rangle, \langle 1_{9996}\,1_{9997}\,1_{9998}\,1_{9999}\rangle\,]$$

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Consider the set $\{9995, 9996, 9997, 9998, 9999\}$.

Before stripping leading zeroes:

$$[\,\underbrace{\langle 0000\rangle, \ldots, \langle 0000\rangle}_{\text{2498 words}}, \langle 000\,1_{9995}\rangle, \langle\, 1_{9996}\,1_{9997}\,1_{9998}\,1_{9999}\,\rangle\,]$$

After stripping leading zeroes:

$$\{\,9992\,[\,\langle 000\,1_{9995}\rangle, \langle\, 1_{9996}\,1_{9997}\,1_{9998}\,1_{9999}\,\rangle\,]\,\}$$

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Consider the set $\{9995, 9996, 9997, 9998, 9999\}$.

Before stripping leading zeroes:

$$[\underbrace{\langle 0000 \rangle, \ldots, \langle 0000 \rangle}_{\text{2498 words}}, \langle 000\,1_{9995} \rangle, \langle 1_{9996}\,1_{9997}\,1_{9998}\,1_{9999} \rangle]$$

After stripping leading zeroes:

$$\{\ 9992\ [\langle 000\,1_{9995} \rangle, \langle 1_{9996}\,1_{9997}\,1_{9998}\,1_{9999} \rangle]\ \}$$

Bit-Vectors: Trailing and Leading Zeroes

Trailing and leading zeroes waste time and space.

- Many implementations strip trailing zeroes by maintaining the array length.
- We further strip leading zeroes by maintaining an offset.

Consider the set $\{9995, 9996, 9997, 9998, 9999\}$.

Before stripping leading zeroes:

$$[\underbrace{\langle 0000\rangle, \ldots, \langle 0000\rangle}_{\text{2498 words}}, \langle 000\, 1_{9995}\rangle, \langle 1_{9996}\, 1_{9997}\, 1_{9998}\, 1_{9999}\rangle]$$

After stripping leading zeroes:

$$\{\, 9992\ [\ \langle 000\, 1_{9995}\rangle, \langle 1_{9996}\, 1_{9997}\, 1_{9998}\, 1_{9999}\rangle\ ]\ \}$$

Representing Points-To Sets with Bit-Vectors

Mapping of memory objects to integral identifiers required.

Representing Points-To Sets with Bit-Vectors

Mapping of memory objects to integral identifiers required.

Consider $pt(p) = \{o_4, o_5, o_{12}, o_{14}\}$.

Map o_n to n: $o_4 \mapsto 4 \quad o_5 \mapsto 5 \quad o_{12} \mapsto 12 \quad o_{14} \mapsto 14$

Representing Points-To Sets with Bit-Vectors

Mapping of memory objects to integral identifiers required.

Consider $pt(p) = \{o_4, o_5, o_{12}, o_{14}\}$.

Map o_n to n: $o_4 \mapsto 4 \quad o_5 \mapsto 5 \quad o_{12} \mapsto 12 \quad o_{14} \mapsto 14$

$$pt(p) = \{\, 4\, [\, \langle 1_4 1_5 0 0 \rangle, \langle 0 0 0 0 \rangle, \langle 1_{12} 0 1_{14} 0 \rangle \,]\, \}$$

Three words required.

Representing Points-To Sets with Bit-Vectors

Mapping of memory objects to integral identifiers required.

Consider $pt(p) = \{o_4, o_5, o_{12}, o_{14}\}$.

Map o_n to n: $o_4 \mapsto 4$ $o_5 \mapsto 5$ $o_{12} \mapsto 12$ $o_{14} \mapsto 14$

$$pt(p) = \{\, 4\, [\, \langle 1_4\, 1_5\, 0\, 0 \rangle, \langle 0\,0\,0\,0 \rangle, \langle 1_{12}\, 0\, 1_{14}\, 0 \rangle \,]\, \}$$

Three words required.

How about... $o_4 \mapsto 4$ $o_5 \mapsto 5$ $o_{12} \mapsto 6$ $o_{14} \mapsto 7$

Representing Points-To Sets with Bit-Vectors

Mapping of memory objects to integral identifiers required.

Consider $pt(p) = \{o_4, o_5, o_{12}, o_{14}\}$.

Map o_n to n: $o_4 \mapsto 4 \quad o_5 \mapsto 5 \quad o_{12} \mapsto 12 \quad o_{14} \mapsto 14$

$$pt(p) = \{\, 4\, [\, \langle 1_4\, 1_5\, 0\, 0 \rangle, \langle 0\,0\,0\,0 \rangle, \langle 1_{12}\, 0\, 1_{14}\, 0 \rangle \,]\, \}$$

Three words required.

How about... $o_4 \mapsto 4 \quad o_5 \mapsto 5 \quad o_{12} \mapsto 6 \quad o_{14} \mapsto 7$

$$pt(p) = \{\, 4\, [\, \langle 1_4\, 1_5\, 1_6\, 1_7 \rangle \,]\, \}$$

One word required.

Representing Points-To Sets with Bit-Vectors

Given a points-to set P with n objects, word size of $\mathcal{W}$, the minimum number of words required to represent P as a bit-vector is

$$\left\lceil \frac{n}{\mathcal{W}} \right\rceil$$

Representing Points-To Sets with Bit-Vectors

Given a points-to set P with n objects, word size of $\mathcal{W}$, the minimum number of words required to represent P as a bit-vector is

$$\left\lceil \frac{n}{\mathcal{W}} \right\rceil$$

Example

$pt(p) = \{o_1, o_2, o_3, o_4, o_5\}$ requires at minimum $\left\lceil \frac{n}{\mathcal{W}} \right\rceil = \left\lceil \frac{5}{4} \right\rceil = 2$ words to represent.

Representing Points-To Sets with Bit-Vectors

Given a points-to set P with n objects, word size of $\mathcal{W}$, the minimum number of words required to represent P as a bit-vector is

$$\left\lceil \frac{n}{\mathcal{W}} \right\rceil$$

Example

$pt(p) = \{o_1, o_2, o_3, o_4, o_5\}$ requires at minimum $\left\lceil \frac{n}{\mathcal{W}} \right\rceil = \left\lceil \frac{5}{4} \right\rceil = 2$ words to represent.

Observation: Objects in same points-to sets should be mapped to numerically close identifiers.

Representing Points-To Sets with Bit-Vectors

Given a points-to set P with n objects, word size of $\mathcal{W}$, the minimum number of words required to represent P as a bit-vector is

$$\left\lceil \frac{n}{\mathcal{W}} \right\rceil$$

Example

$pt(p) = \{o_1, o_2, o_3, o_4, o_5\}$ requires at minimum $\left\lceil \frac{n}{\mathcal{W}} \right\rceil = \left\lceil \frac{5}{4} \right\rceil = 2$ words to represent.

Observation: Objects in same points-to sets should be mapped to numerically close identifiers.

Auxiliary analysis soundly over-approximates main analysis...

$$\text{Good mapping for auxiliary analysis} \xrightarrow{\textit{probably}} \text{Good mapping for main analysis.}$$

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

For each points-to set $P = \{o_{x_1}, o_{x_2}, \ldots, o_{x_n}\}$ produced by the auxiliary analysis, we have the following known variables:

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

For each points-to set $P = \{o_{x_1}, o_{x_2}, \ldots, o_{x_n}\}$ produced by the auxiliary analysis, we have the following known variables:

1. n is the number of objects in P.

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

For each points-to set $P = \{o_{x_1}, o_{x_2}, \ldots, o_{x_n}\}$ produced by the auxiliary analysis, we have the following known variables:

1. n is the number of objects in P.
2. $w = \left\lceil \frac{n}{\mathcal{W}} \right\rceil$ is the minimum number of words required for a bit-vector representation of P.

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

For each points-to set $P = \{o_{x_1}, o_{x_2}, \ldots, o_{x_n}\}$ produced by the auxiliary analysis, we have the following known variables:

1. n is the number of objects in P.
2. $w = \left\lceil \frac{n}{\mathcal{W}} \right\rceil$ is the minimum number of words required for a bit-vector representation of P.

And the following unknown variables:

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

For each points-to set $P = \{o_{x_1}, o_{x_2}, \ldots, o_{x_n}\}$ produced by the auxiliary analysis, we have the following known variables:

1. n is the number of objects in P.
2. $w = \left\lceil \frac{n}{\mathcal{W}} \right\rceil$ is the minimum number of words required for a bit-vector representation of P.

And the following unknown variables:

1. m_{x_i}, where $1 \leq i \leq n$, is the identifier object o_{x_i} will be assigned to in our new mapping ($o_{x_i} \mapsto m_{x_i}$).

Integer Programming Formulation

Optimisation problem – integer programming can give us an optimal solution.

For each points-to set $P = \{o_{x_1}, o_{x_2}, \ldots, o_{x_n}\}$ produced by the auxiliary analysis, we have the following known variables:

1. n is the number of objects in P.
2. $w = \left\lceil \frac{n}{\mathcal{W}} \right\rceil$ is the minimum number of words required for a bit-vector representation of P.

And the following unknown variables:

1. m_{x_i}, where $1 \leq i \leq n$, is the identifier object o_{x_i} will be assigned to in our new mapping $(o_{x_i} \mapsto m_{x_i})$.
2. f is some offset multiplier for where the identifiers, m_{x_i}, start.

Integer Programming Formulation

For each points-to set P:

$$m_{x_i} \geq f_P \cdot \mathcal{W} \tag{C1}$$

Integer Programming Formulation

For each points-to set P:

$$\begin{aligned} m_{x_i} &\geq f_P \cdot \mathcal{W} \\ m_{x_i} &< f_P \cdot \mathcal{W} + w_P \cdot \mathcal{W} \end{aligned} \tag{C1}$$

Integer Programming Formulation

For each points-to set P:

$$\begin{aligned} m_{x_i} &\geq f_P \cdot \mathcal{W} \\ m_{x_i} &< f_P \cdot \mathcal{W} + w_P \cdot \mathcal{W} \\ f_P &\geq 0 \end{aligned} \tag{C1}$$

Integer Programming Formulation

For each points-to set P:

$$\begin{aligned} m_{x_i} &\geq f_P \cdot \mathcal{W} \\ m_{x_i} &< f_P \cdot \mathcal{W} + w_P \cdot \mathcal{W} \\ f_P &\geq 0 \end{aligned} \tag{C1}$$

Whilst also ensuring the mapping is 1-to-1; for each i, j, $i \neq j$:

$$\left| m_i - m_j \right| > 0 \tag{C2}$$

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$

- o_1 should be close to o_2, o_3, and o_4

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$

- o_1 should be close to o_2, o_3, and o_4
- o_2 should be close to o_1 and o_5.

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{\boxed{o_1}, \boxed{o_2}\}$, $P_2 = \{\boxed{o_1}, o_3, o_4\}$, $P_3 = \{\boxed{o_2}, o_5\}$

- o_1 should be close to o_2, o_3, and o_4
- o_2 should be close to o_1 and o_5.

With $\mathcal{W} = 4$, $w = 1$ for all 3 sets, but:

- The sets overlap.

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$

- o_1 should be close to o_2, o_3, and o_4
- o_2 should be close to o_1 and o_5.

With $\mathcal{W} = 4$, $w = 1$ for all 3 sets, but:

- The sets overlap.

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$

- o_1 should be close to o_2, o_3, and o_4
- o_2 should be close to o_1 and o_5.

With $\mathcal{W} = 4$, $w = 1$ for all 3 sets, but:

- The sets overlap.
- There are 5 objects in total.

Pigeonhole Principle

Definition

Pigeonhole principle. Given n items to be spread amongst m containers, $n > m$, at least one container must contain more than one item.

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$

- o_1 should be close to o_2, o_3, and o_4
- o_2 should be close to o_1 and o_5.

With $\mathcal{W} = 4$, $w = 1$ for all 3 sets, but:

- The sets overlap.
- There are 5 objects in total.

Pigeonhole principle prevents a perfect solution

Integer Programming Formulation

For each points-to set P:

$$\begin{aligned} m_{x_i} &\geq f_P \cdot \mathcal{W} \\ m_{x_i} &< f_P \cdot \mathcal{W} + w_P \cdot \mathcal{W} \\ f_P &\geq 0 \end{aligned} \tag{C1}$$

Whilst also ensuring the mapping is 1-to-1; for each i, j, $i \neq j$:

$$\left| m_i - m_j \right| > 0 \tag{C2}$$

Integer Programming Formulation

For each points-to set P:

$$\begin{aligned} m_{x_i} &\geq f_P \cdot \mathcal{W} \\ m_{x_i} &< f_P \cdot \mathcal{W} + w_P \cdot \mathcal{W} + \boxed{t_P \cdot \mathcal{W}} \\ f_P &\geq 0 \end{aligned} \tag{C1}$$

Whilst also ensuring the mapping is 1-to-1; for each i, j, $i \neq j$:

$$\left| m_i - m_j \right| > 0 \tag{C2}$$

Optimise for minimum tolerances: $t_{P_1} + \cdots + t_{P_n}$.

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$m_1 \geq 4 \cdot f_{P_1}$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$m_1 \geq 4 \cdot f_{P_1}$$
$$m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \frac{2}{4} \right\rceil + 4 \cdot t_{P_1}$$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$m_1 \geq 4 \cdot f_{P_1}$
$m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \frac{2}{4} \right\rceil + 4 \cdot t_{P_1}$
$m_2 \geq 4 \cdot f_{P_1}$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$
\begin{aligned}
&m_1 \geq 4 \cdot f_{P_1} \\
&m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
&m_2 \geq 4 \cdot f_{P_1} \\
&m_2 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1}
\end{aligned}
$$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$
\begin{aligned}
&m_1 \geq 4 \cdot f_{P_1}\\
&m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1}\\
&m_2 \geq 4 \cdot f_{P_1}\\
&m_2 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1}\\
&f_{P_1} \geq 0
\end{aligned}
$$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$
\begin{aligned}
& m_1 \geq 4 \cdot f_{P_1} \\
& m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
& m_2 \geq 4 \cdot f_{P_1} \\
& m_2 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
& f_{P_1} \geq 0
\end{aligned}
\qquad
\begin{aligned}
& m_1 \geq 4 \cdot f_{P_2} \\
& m_1 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
& m_3 \geq 4 \cdot f_{P_2} \\
& m_3 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
& m_5 \geq 4 \cdot f_{P_2} \\
& m_5 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
& f_{P_2} \geq 0
\end{aligned}
$$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$
\begin{aligned}
m_1 &\geq 4 \cdot f_{P_1} \\
m_1 &< 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
m_2 &\geq 4 \cdot f_{P_1} \\
m_2 &< 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
f_{P_1} &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
m_1 &\geq 4 \cdot f_{P_2} \\
m_1 &< 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
m_3 &\geq 4 \cdot f_{P_2} \\
m_3 &< 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
m_5 &\geq 4 \cdot f_{P_2} \\
m_5 &< 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
f_{P_2} &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
m_2 &\geq 4 \cdot f_{P_3} \\
m_2 &< 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
m_5 &\geq 4 \cdot f_{P_3} \\
m_5 &< 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
f_{P_3} &\geq 0
\end{aligned}
$$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$
\begin{aligned}
& m_1 \geq 4 \cdot f_{P_1} \\
& m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
& m_2 \geq 4 \cdot f_{P_1} \\
& m_2 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
& f_{P_1} \geq 0
\end{aligned}
$$

$$
\begin{aligned}
& m_1 \geq 4 \cdot f_{P_2} \\
& m_1 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
& m_3 \geq 4 \cdot f_{P_2} \\
& m_3 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
& m_5 \geq 4 \cdot f_{P_2} \\
& m_5 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
& f_{P_2} \geq 0
\end{aligned}
$$

$$
\begin{aligned}
& m_2 \geq 4 \cdot f_{P_3} \\
& m_2 < 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
& m_5 \geq 4 \cdot f_{P_3} \\
& m_5 < 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
& f_{P_3} \geq 0
\end{aligned}
$$

$$
\begin{gathered}
|m_1 - m_2| > 0,\ |m_1 - m_3| > 0,\ |m_1 - m_4| > 0,\ |m_1 - m_5| > 0,\ |m_2 - m_3| > 0, \\
|m_2 - m_4| > 0,\ |m_2 - m_5| > 0,\ |m_3 - m_4| > 0,\ |m_3 - m_5| > 0,\ |m_4 - m_5| > 0
\end{gathered}
$$

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$
\begin{aligned}
&m_1 \geq 4 \cdot f_{P_1} \\
&m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
&m_2 \geq 4 \cdot f_{P_1} \\
&m_2 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
&f_{P_1} \geq 0
\end{aligned}
$$

$$
\begin{aligned}
&m_1 \geq 4 \cdot f_{P_2} \\
&m_1 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
&m_3 \geq 4 \cdot f_{P_2} \\
&m_3 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
&m_5 \geq 4 \cdot f_{P_2} \\
&m_5 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
&f_{P_2} \geq 0
\end{aligned}
$$

$$
\begin{aligned}
&m_2 \geq 4 \cdot f_{P_3} \\
&m_2 < 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
&m_5 \geq 4 \cdot f_{P_3} \\
&m_5 < 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
&f_{P_3} \geq 0
\end{aligned}
$$

$$
\begin{gathered}
|m_1 - m_2| > 0,\ |m_1 - m_3| > 0,\ |m_1 - m_4| > 0,\ |m_1 - m_5| > 0,\ |m_2 - m_3| > 0, \\
|m_2 - m_4| > 0,\ |m_2 - m_5| > 0,\ |m_3 - m_4| > 0,\ |m_3 - m_5| > 0,\ |m_4 - m_5| > 0
\end{gathered}
$$

Optimise for minimum $t_{P_1} + t_{P_2} + t_{P_3}$.

Integer Programming Formulation

Consider $P_1 = \{o_1, o_2\}$, $P_2 = \{o_1, o_3, o_4\}$, $P_3 = \{o_2, o_5\}$, $\mathcal{W} = 4$

$$\begin{aligned}
&m_1 \geq 4 \cdot f_{P_1} \\
&m_1 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
&m_2 \geq 4 \cdot f_{P_1} \\
&m_2 < 4 \cdot f_{P_1} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_1} \\
&f_{P_1} \geq 0
\end{aligned}$$

$$\begin{aligned}
&m_1 \geq 4 \cdot f_{P_2} \\
&m_1 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
&m_3 \geq 4 \cdot f_{P_2} \\
&m_3 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
&m_5 \geq 4 \cdot f_{P_2} \\
&m_5 < 4 \cdot f_{P_2} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_2} \\
&f_{P_2} \geq 0
\end{aligned}$$

$$\begin{aligned}
&m_2 \geq 4 \cdot f_{P_3} \\
&m_2 < 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
&m_5 \geq 4 \cdot f_{P_3} \\
&m_5 < 4 \cdot f_{P_3} + 4 \cdot \left\lceil \tfrac{2}{4} \right\rceil + 4 \cdot t_{P_3} \\
&f_{P_3} \geq 0
\end{aligned}$$

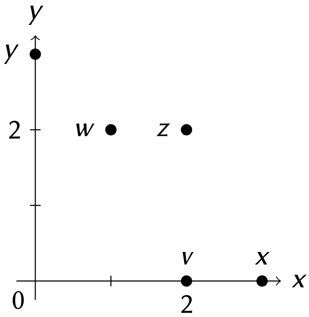
$$|m_1 - m_2| > 0,\ |m_1 - m_3| > 0,\ |m_1 - m_4| > 0,\ |m_1 - m_5| > 0,\ |m_2 - m_3| > 0,$$
$$|m_2 - m_4| > 0,\ |m_2 - m_5| > 0,\ |m_3 - m_4| > 0,\ |m_3 - m_5| > 0,\ |m_4 - m_5| > 0$$

Optimise for minimum $t_{P_1} + t_{P_2} + t_{P_3}$.

Optimal... but costly! Let's try something more approximate...

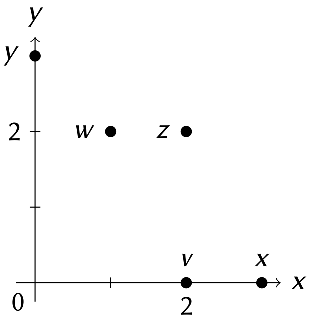
Hierarchical Clustering

Hierarchical Clustering



Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

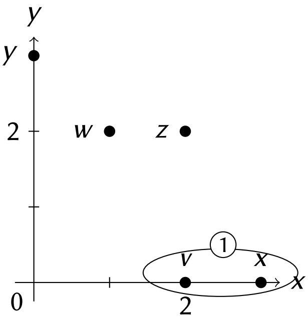
Hierarchical Clustering



Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

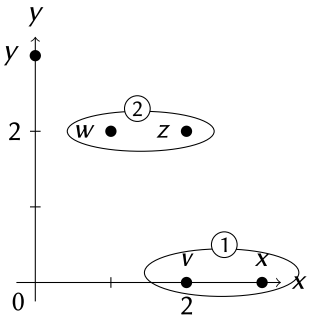
Linkage criterion: single linkage – distance between two clusters is the smallest distance between any member of one cluster to any member of the other.

Hierarchical Clustering



Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
Linkage criterion: single linkage – distance between two clusters is the smallest distance between any member of one cluster to any member of the other.

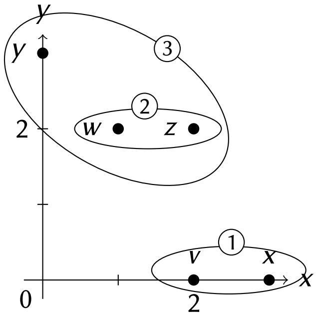
Hierarchical Clustering



Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

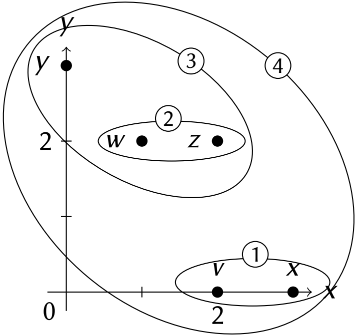
Linkage criterion: single linkage – distance between two clusters is the smallest distance between any member of one cluster to any member of the other.

Hierarchical Clustering



Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
Linkage criterion: single linkage – distance between two clusters is the smallest distance between any member of one cluster to any member of the other.

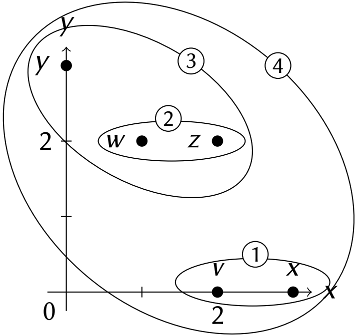
Hierarchical Clustering

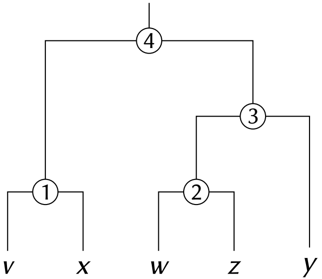


Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Linkage criterion: single linkage – distance between two clusters is the smallest distance between any member of one cluster to any member of the other.

Hierarchical Clustering





Distance function: Euclidean distance $-\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
Linkage criterion: single linkage – distance between two clusters is the smallest distance between any member of one cluster to any member of the other.

Clustering Objects

Clustering Objects

Definition

Object Distance.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.
2. Choose a linkage criterion and perform hierarchical clustering.
    - Produces dendrogram.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.
2. Choose a linkage criterion and perform hierarchical clustering.
   - Produces dendrogram.
3. Initialise a counter c to 0.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.
2. Choose a linkage criterion and perform hierarchical clustering.
   - Produces dendrogram.
3. Initialise a counter c to 0.
4. Perform depth-first search on dendrogram.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.
2. Choose a linkage criterion and perform hierarchical clustering.
   - Produces dendrogram.
3. Initialise a counter c to 0.
4. Perform depth-first search on dendrogram.
   - At each leaf containing object o.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.
2. Choose a linkage criterion and perform hierarchical clustering.
   - Produces dendrogram.
3. Initialise a counter c to 0.
4. Perform depth-first search on dendrogram.
   - At each leaf containing object o.
     1. Map o to c.

Clustering Objects

Definition

Object Distance. The distance between two objects is the minimum number of words required to represent any points-to set in which both objects appear in as a bit-vector.

1. Build a distance matrix: $dmat[o_1][o_2]$ holds the distance between o_1 and o_2.
2. Choose a linkage criterion and perform hierarchical clustering.
   - Produces dendrogram.
3. Initialise a counter c to 0.
4. Perform depth-first search on dendrogram.
   - At each leaf containing object o.
     1. Map o to c.
     2. Increment c.

Optimisation: Region-Based Clustering

Optimisation: Region-Based Clustering

If two objects do not appear in a points-to set together, their mappings do not affect each other.

- Their distance is ∞.

Optimisation: Region-Based Clustering

If two objects do not appear in a points-to set together, their mappings do not affect each other.

- Their distance is ∞.

We can cluster *regions* of objects which depend on each other separately.

Optimisation: Region-Based Clustering

If two objects do not appear in a points-to set together, their mappings do not affect each other.

- Their distance is ∞.

We can cluster *regions* of objects which depend on each other separately.

- Save time: cluster concurrently (not really necessary).

Optimisation: Region-Based Clustering

If two objects do not appear in a points-to set together, their mappings do not affect each other.

- Their distance is ∞.

We can cluster *regions* of objects which depend on each other separately.

- Save time: cluster concurrently (not really necessary).
- Save space: distance matrix is quadratic in size.

Optimisation: Region-Based Clustering

If two objects do not appear in a points-to set together, their mappings do not affect each other.

- Their distance is ∞.

We can cluster *regions* of objects which depend on each other separately.

- Save time: cluster concurrently (not really necessary).
- Save space: distance matrix is quadratic in size.

Regions with fewer than $\mathcal{W}$ objects can have their objects mapped arbitrarily.

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Consider two regions R_1 and R_2

$$R_1 : o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2$$
$$R_2 : o_d \mapsto 3 \quad o_e \mapsto 4 \quad o_f \mapsto 5$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Consider two regions R_1 and R_2

$$R_1 : o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2$$
$$R_2 : o_d \mapsto 3 \quad o_e \mapsto 4 \quad o_f \mapsto 5$$

Any points-to set in R_1 (subsets of $\{o_a, o_b, o_c\}$) will take the form

$$\{\, 0\,[\,\langle \times\times\times 0\rangle\,]\,\},$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Consider two regions R_1 and R_2

$$R_1 : o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2$$
$$R_2 : o_d \mapsto 3 \quad o_e \mapsto 4 \quad o_f \mapsto 5$$

Any points-to set in R_1 (subsets of $\{o_a, o_b, o_c\}$) will take the form

$$\{\, 0\, [\, \langle \times \times \times 0 \rangle \,] \,\},$$

Any points-to set in R_2 (subsets of $\{o_d, o_e, o_f\}$) will take one of three forms

$$\begin{aligned} \{\, 0\, [\, \langle 0001 \rangle \,] \,\} &\quad \{o_d\} \\ \{\, 4\, [\, \langle \times \times 00 \rangle \,] \,\} &\quad \{o_e\}, \{o_f\}, \{o_e, o_f\} \\ \{\, 0\, [\, \langle 0001 \rangle, \langle \times \times 00 \rangle \,] \,\} &\quad \{o_d, o_e\}, \{o_d, o_f\}, \{o_d, o_e, o_f\} \end{aligned}$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Consider two regions R_1 and R_2

$$R_1 : o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2$$
$$R_2 : o_d \mapsto 3 \quad o_e \mapsto 4 \quad o_f \mapsto 5$$

Any points-to set in R_1 (subsets of $\{o_a, o_b, o_c\}$) will take the form

$$\{\, 0\, [\, \langle \times \times \times 0 \rangle \,]\, \},$$

Any points-to set in R_2 (subsets of $\{o_d, o_e, o_f\}$) will take one of three forms

$$\{\, 0\, [\, \langle 0001 \rangle \,]\, \} \quad \{o_d\}$$
$$\{\, 4\, [\, \langle \times \times 00 \rangle \,]\, \} \quad \{o_e\}, \{o_f\}, \{o_e, o_f\}$$
$$\{\, 0\, [\, \langle 0001 \rangle, \langle \times \times 00 \rangle \,]\, \} \quad \{o_d, o_e\}, \{o_d, o_f\}, \{o_d, o_e, o_f\}$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Instead, consider the mapping

$$
\begin{aligned}
R_1 &: o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2 \\
R_2 &: o_d \mapsto 4 \quad o_e \mapsto 5 \quad o_f \mapsto 6
\end{aligned}
$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Instead, consider the mapping

$$
\begin{array}{llll}
R_1 : & o_a \mapsto \boxed{0} & o_b \mapsto 1 & o_c \mapsto 2 \\
R_2 : & o_d \mapsto \boxed{4} & o_e \mapsto 5 & o_f \mapsto 6
\end{array}
$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Instead, consider the mapping

$$R_1 : o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2$$
$$R_2 : o_d \mapsto 4 \quad o_e \mapsto 5 \quad o_f \mapsto 6$$

As before, any points-to set with objects in R_1 will take the form

$$\{\, 0 \,[\, \langle \times \times \times 0 \rangle \,]\, \}$$

Optimisation: Word-Aligned Identifier Mapping

The first identifier in a region should always start at a word-aligned boundary.

Example

Instead, consider the mapping

$$R_1 : o_a \mapsto 0 \quad o_b \mapsto 1 \quad o_c \mapsto 2$$
$$R_2 : o_d \mapsto 4 \quad o_e \mapsto 5 \quad o_f \mapsto 6$$

As before, any points-to set with objects in R_1 will take the form

$$\{\, 0\, [\, \langle \times \times \times 0 \rangle \,]\, \}$$

But now, any points-to set with objects in R_2 will take the form

$$\{\, 4\, [\, \langle \times \times \times 0 \rangle \,]\, \}$$

Implementation

- Implemented in LLVM-based points-to analysis framework SVF.
  - SFS algorithm unchanged.
  - Flow-insensitive Andersen's analysis as auxiliary analysis.

Implementation

- Implemented in LLVM-based points-to analysis framework SVF.
  - SFS algorithm unchanged.
  - Flow-insensitive Andersen's analysis as auxiliary analysis.
- *fastcluster*[1] with *hclust-cpp* [2] C++ bindings for clustering.

[1] Daniel Müllner, fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python, Journal of Statistical Software 53 (2013), no. 9, 1–18, URL http://www.jstatsoft.org/v53/i09/

[2] https://github.com/cdalitz/hclust-cpp

Implementation

- Implemented in LLVM-based points-to analysis framework SVF.
  - SFS algorithm unchanged.
  - Flow-insensitive Andersen's analysis as auxiliary analysis.
- *fastcluster*[1] with *hclust-cpp*[2] C++ bindings for clustering.
- Clustered with 3 linkage criteria – single, complete, and average – and chose the best for the auxiliary analysis.

[1] Daniel Müllner, fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python, Journal of Statistical Software 53 (2013), no. 9, 1–18, URL http://www.jstatsoft.org/v53/i09/

[2] https://github.com/cdalitz/hclust-cpp

Implementation

- Implemented in LLVM-based points-to analysis framework SVF.
  - SFS algorithm unchanged.
  - Flow-insensitive Andersen's analysis as auxiliary analysis.
- *fastcluster*[1] with *hclust-cpp* [2] C++ bindings for clustering.
- Clustered with 3 linkage criteria – single, complete, and average – and chose the best for the auxiliary analysis.
- Word size of 64.

[1] Daniel Müllner, fastcluster: Fast Hierarchical, Agglomerative Clustering Routines for R and Python, Journal of Statistical Software 53 (2013), no. 9, 1–18, URL http://www.jstatsoft.org/v53/i09/

[2] https://github.com/cdalitz/hclust-cpp

Evaluation: Word Reduction

Benchmark	**Theoretical**	**Original**	**Single**	**Complete**	**Average**	**Reduction**
dhcpcd	3 317 195	23 911 465	**4 961 417**	6 605 816	5 784 023	4.82 ×
gawk	58 007 460	429 739 789	**82 783 110**	140 588 641	148 836 214	5.19 ×
bash	26 586 881	295 168 808	**31 731 607**	36 861 568	47 120 912	9.30 ×
mutt	51 298 142	548 971 273	**87 213 543**	260 457 927	259 746 461	6.29 ×
lynx	133 664 618	1 015 676 964	**237 113 529**	289 849 510	302 122 259	4.28 ×
xpdf	731 879 787	4 197 513 654	1 558 434 196	1 558 496 134	**1 526 729 185**	2.75 ×
ruby	320 059 196	6 600 730 356	**1 405 659 097**	2 514 836 137	2 186 425 117	4.70 ×
keepassxc	13 770 856	1 399 786 369	**107 456 539**	134 257 502	120 881 288	13.03×
Geo. Mean						5.66×

(**Theoretical** is $\lceil \frac{n}{\mathcal{W}} \rceil$.)

Evaluation: Word Reduction

Benchmark	**Theoretical**	**Original**	**Single**	**Complete**	**Average**	**Reduction**
dhcpcd	3 317 195	23 911 465	**4 961 417**	6 605 816	5 784 023	4.82 ×
gawk	58 007 460	429 739 789	**82 783 110**	140 588 641	148 836 214	5.19 ×
bash	26 586 881	295 168 808	**31 731 607**	36 861 568	47 120 912	9.30 ×
mutt	51 298 142	548 971 273	**87 213 543**	260 457 927	259 746 461	6.29 ×
lynx	133 664 618	1 015 676 964	**237 113 529**	289 849 510	302 122 259	4.28 ×
xpdf	731 879 787	4 197 513 654	1 558 434 196	1 558 496 134	**1 526 729 185**	2.75 ×
ruby	320 059 196	6 600 730 356	**1 405 659 097**	2 514 836 137	2 186 425 117	4.70 ×
keepassxc	13 770 856	1 399 786 369	**107 456 539**	134 257 502	120 881 288	13.03 ×
Geo. Mean						5.66 ×

(**Theoretical** is $\left\lceil \frac{n}{\mathcal{W}} \right\rceil$.)

Evaluation: Word Reduction

Benchmark	**Theoretical**	**Original**	**Single**	**Complete**	**Average**	**Reduction**
dhcpcd	3 317 195	23 911 465	**4 961 417**	6 605 816	5 784 023	4.82 ×
gawk	58 007 460	429 739 789	**82 783 110**	140 588 641	148 836 214	5.19 ×
bash	26 586 881	295 168 808	**31 731 607**	36 861 568	47 120 912	9.30 ×
mutt	51 298 142	548 971 273	**87 213 543**	260 457 927	259 746 461	6.29 ×
lynx	133 664 618	1 015 676 964	**237 113 529**	289 849 510	302 122 259	4.28 ×
xpdf	731 879 787	4 197 513 654	1 558 434 196	1 558 496 134	**1 526 729 185**	2.75 ×
ruby	320 059 196	6 600 730 356	**1 405 659 097**	2 514 836 137	2 186 425 117	4.70 ×
keepassxc	13 770 856	1 399 786 369	**107 456 539**	134 257 502	120 881 288	13.03×
Geo. Mean						5.66×

(**Theoretical** is $\left\lceil \frac{n}{\mathcal{W}} \right\rceil$.)

Speedup: 1.32×

Evaluation: Word Reduction

Benchmark	Theoretical	Original	Single	Complete	Average	Reduction
dhcpcd	3 317 195	23 911 465	**4 961 417**	6 605 816	5 784 023	4.82 ×
gawk	58 007 460	429 739 789	**82 783 110**	140 588 641	148 836 214	5.19 ×
bash	26 586 881	295 168 808	**31 731 607**	36 861 568	47 120 912	9.30 ×
mutt	51 298 142	548 971 273	**87 213 543**	260 457 927	259 746 461	6.29 ×
lynx	133 664 618	1 015 676 964	**237 113 529**	289 849 510	302 122 259	4.28 ×
xpdf	731 879 787	4 197 513 654	1 558 434 196	1 558 496 134	**1 526 729 185**	2.75 ×
ruby	320 059 196	6 600 730 356	**1 405 659 097**	2 514 836 137	2 186 425 117	4.70 ×
keepassxc	13 770 856	1 399 786 369	**107 456 539**	134 257 502	120 881 288	13.03×
Geo. Mean						5.66×

(**Theoretical** is $\left\lceil \frac{n}{\mathcal{W}} \right\rceil$.)

Speedup: 1.32×

Memory reduction: ≥2.35×

Thank you

Implementation available at https://github.com/SVF-tools/SVF/wiki/Object-Clustering