Finding Optimal Compatible Set of Software Components Using Integer Linear Programming

Jakub Danek, Premek Brada
University of West Bohemia
Pilsen, Czech Republic

Preliminaries

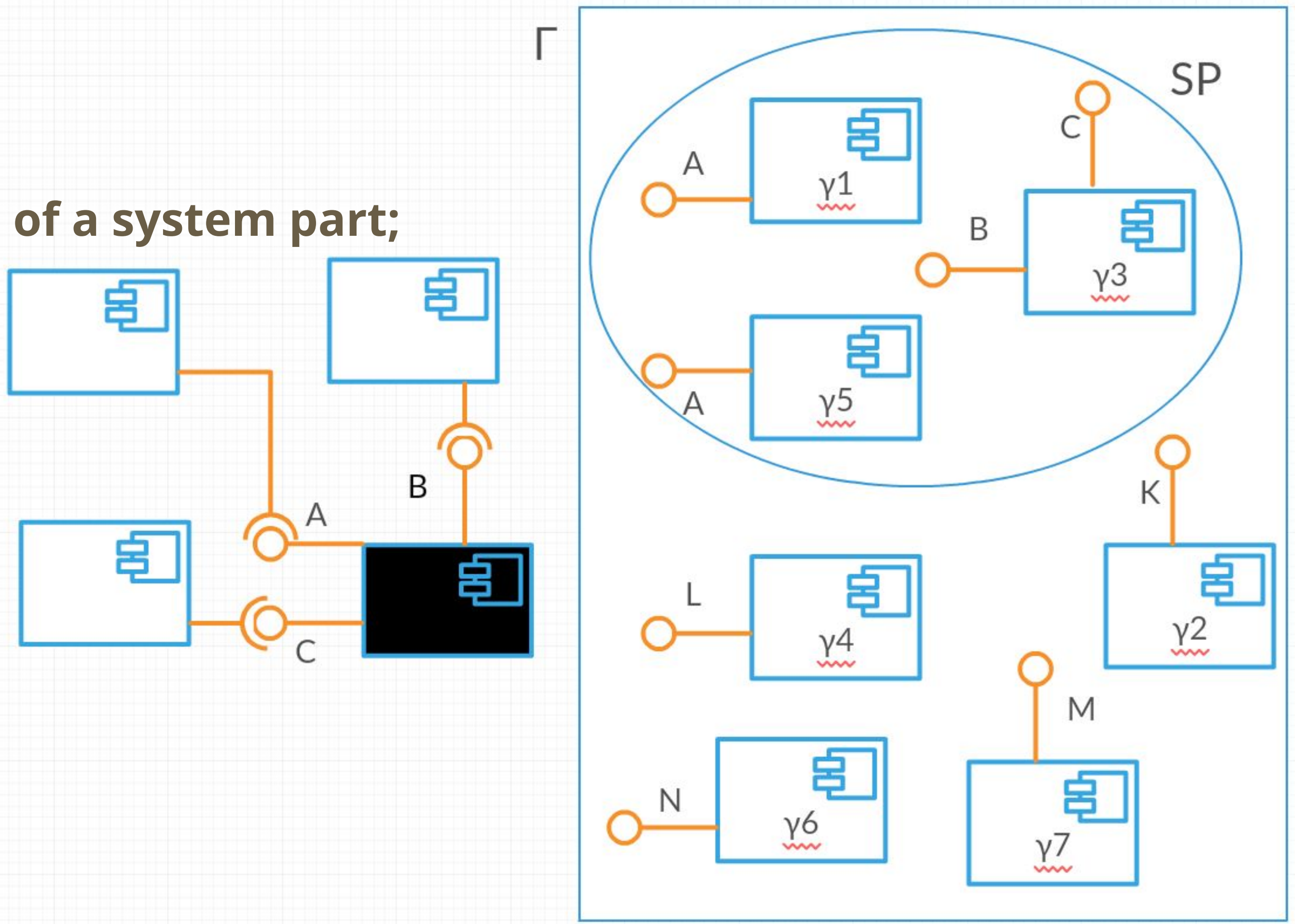
What is a component?

- "A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to third-party composition." (C. Szyperski)

- distributable module, library (e.g. jar file)

Use Cases

1. **Safe upgrade of a system part;**

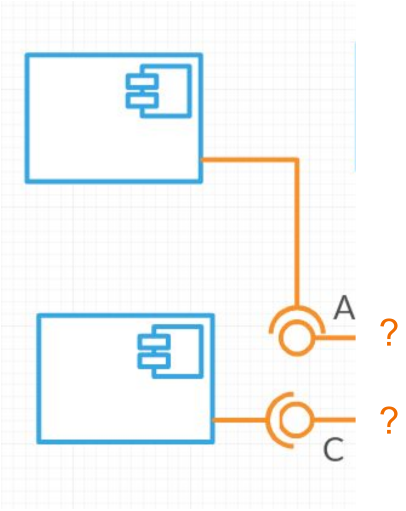


Use Cases

1. Safe upgrade of a system part;

2. **Finding a missing dependency ;**
    - input: set of requirements;
    - context help in IDEs

3. Removing incompatible library duplicates;



Use Cases

1. Safe upgrade of a system portion;

2. Finding a missing dependency;

3. **Removing incompatible library duplicates;**
    - Analysis of corpus of about 600 Java programs (Qualitas Corpus: Tempero et al 2010);

      → open-source, including popular: hibernate, antlr, maven;
      → 75% contain compatibility problem (multiple versions of the same library on classpath);

Goals

1. Finding a set of components that
   1. have particular capabilities such as API (given as user input);

   2. is optimized in respect to an arbitrary metrics;
      - i.e. finding the “best” answer;

2. Achieving both goals “fast enough” for common use during development;
   → e.g. intellisense in IDE, regular evaluation during build, etc.

3. No need to enhance existing applications by custom model;

Related Work

- integration testing
- Desnos et al. 2008 - 1:M replacement, custom model
- Olaechea et al. 2014 - multi-objective sw. product lines

Integer Linear Programming

- search and optimization
- variables (decisions)
- constraints
- objective function **c(x)**

Problem Model

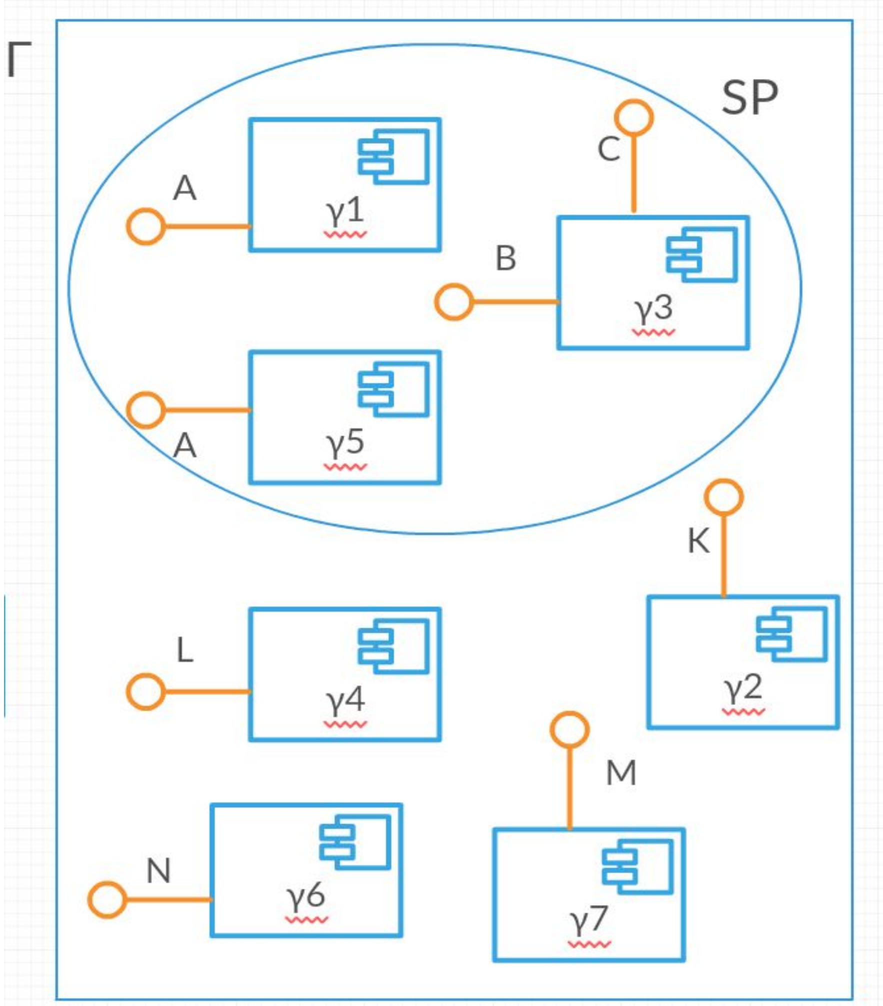
Definitions:

Γ - set of all existing components

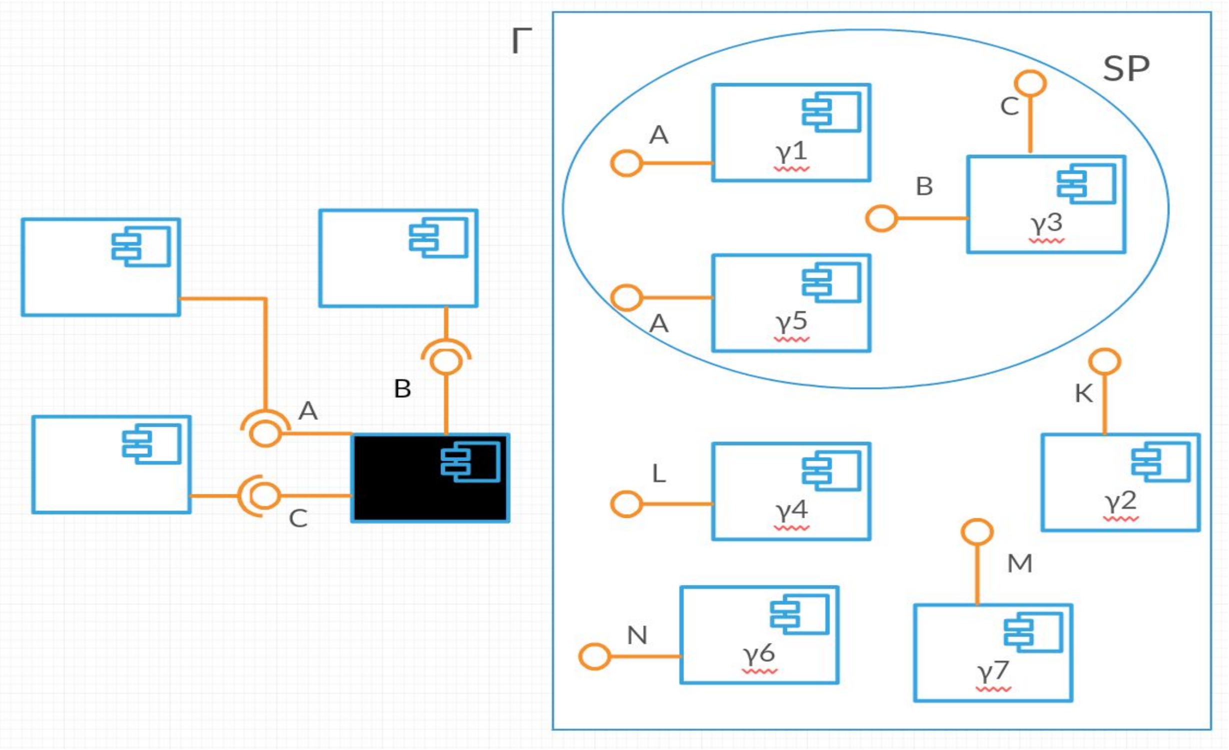
R = {A,B,C} - set of requirements

SP = {γ_1, γ_3, γ_5} - set of components which satisfy at least a single requirement

→ **SP ⊆ Γ**



Problem Model



Definitions:

Γ - set of all existing components

R = {A,B,C} - set of requirements

SP = {γ_1, γ_3, γ_5} - set of components which satisfy at least a single requirement ("search space")

ILP Model

Canonical form:

$\min c^T x$ or $\max c^T x$

$Ax = \{1\}^{|R|}$

$\forall i: x_i \in \{0, 1\}$

ILP Model

Canonical form:

$\min c^T\mathbf{x}$ or $\max c^T\mathbf{x}$

$A\mathbf{x} = \{1\}^{|R|}$

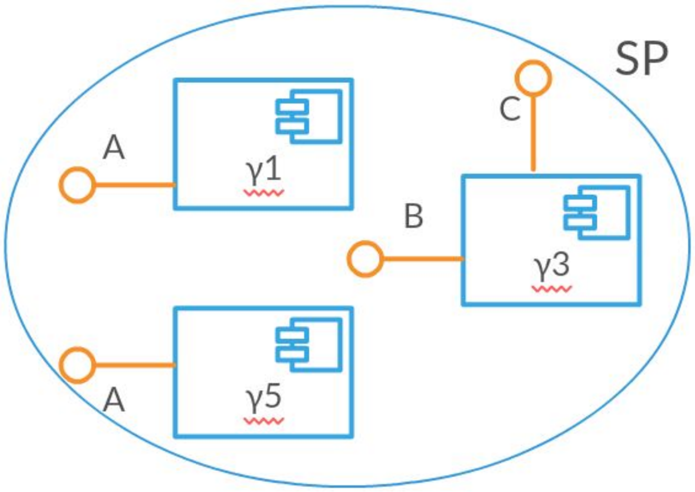
$\forall i: \mathbf{x}_i \in \{0, 1\}$

x - solution vector representing components from the set **SP**

$|x| = |SP|$

1 - the component is part of solution

0 - otherwise



$x = \{\gamma_1, \gamma_3, \gamma_5\}$

ILP Model

Canonical form:

$\min c^T x$ or $\max c^T x$

$\mathbf{Ax = \{1\}^{|R|}}$

$\forall i: x_i \in \{0, 1\}$

A - matrix $|R| \times |x|$

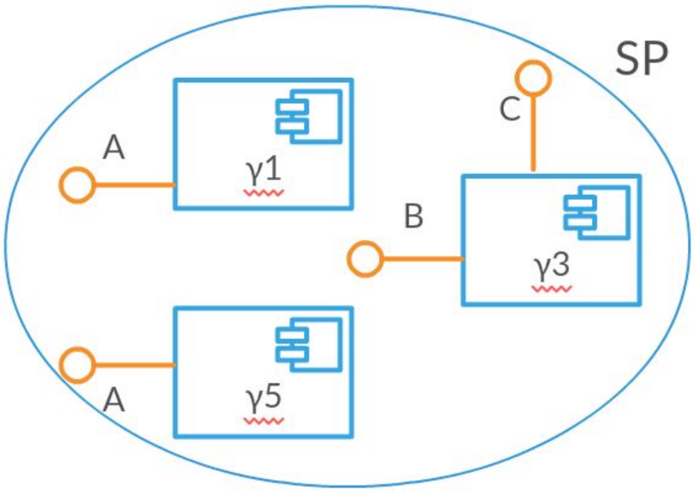
→ rows - requirements

→ columns - components

→ $\forall i,j: a_{ij} \in \{0, 1\}$

1 - the requirement **i** is satisfied by the component **j**

0 - otherwise



	γ_1	γ_3	γ_5
A	1	0	1
B	0	1	0
C	0	1	0

ILP Model

Canonical form:

$$\min\ \mathbf{c}^T x \text{ or } \max\ \mathbf{c}^T x$$

$$Ax = \{1\}^{|R|}$$

$$\forall i:\ x_i \in \{0, 1\}$$

c - vector of component costs

→ optimization function (`min` or `max`)

depends on nature of the cost function

→ size vs benchmark score

ILP Model

Canonical form:

$\min c^T x$ or $\max c^T x$

$Ax = \{1\}^{|R|}$

$\forall i: x_i \in \{0, 1\}$

x - solution vector representing components from the set **SP**

$|x| = |SP|$

1 - the component is part of solution

0 - otherwise

A - matrix $|R| \times |x|$

→ rows - requirements

→ columns - components

→ $\forall i,j: a_{ij} \in \{0, 1\}$

1 - the requirement **i** is satisfied by the component **j**

0 - otherwise

c - vector of component costs

→ optimization function (`min` or `max`)

depends on nature of the cost function

→ size vs benchmark score

Γ and SP Source

Meta-Data Repository - pre-processed for each component:

- full public API image (capabilities)
- set of its dependencies (requirements)
- various metrics
- language and component model independent

Fast Enough? - Simulation

- artificially generated **SP** and **R** sets and values of the **A** matrix

- based on data from previous research (Jezek, Ambroz 2015);
  → dependency problems found in Apache Lucene library
  dependencies on ant, junit and ant-junit

- two scenarios
  a. optimization heavy - lots of feasible solutions
  b. no-optimization - single feasible solution

Intel Core i7 3612QM (4 cores)
8GB RAM
Gentoo Linux kernel 4.0.5
Solver: *Gurobi 6.0.4*

Simulation

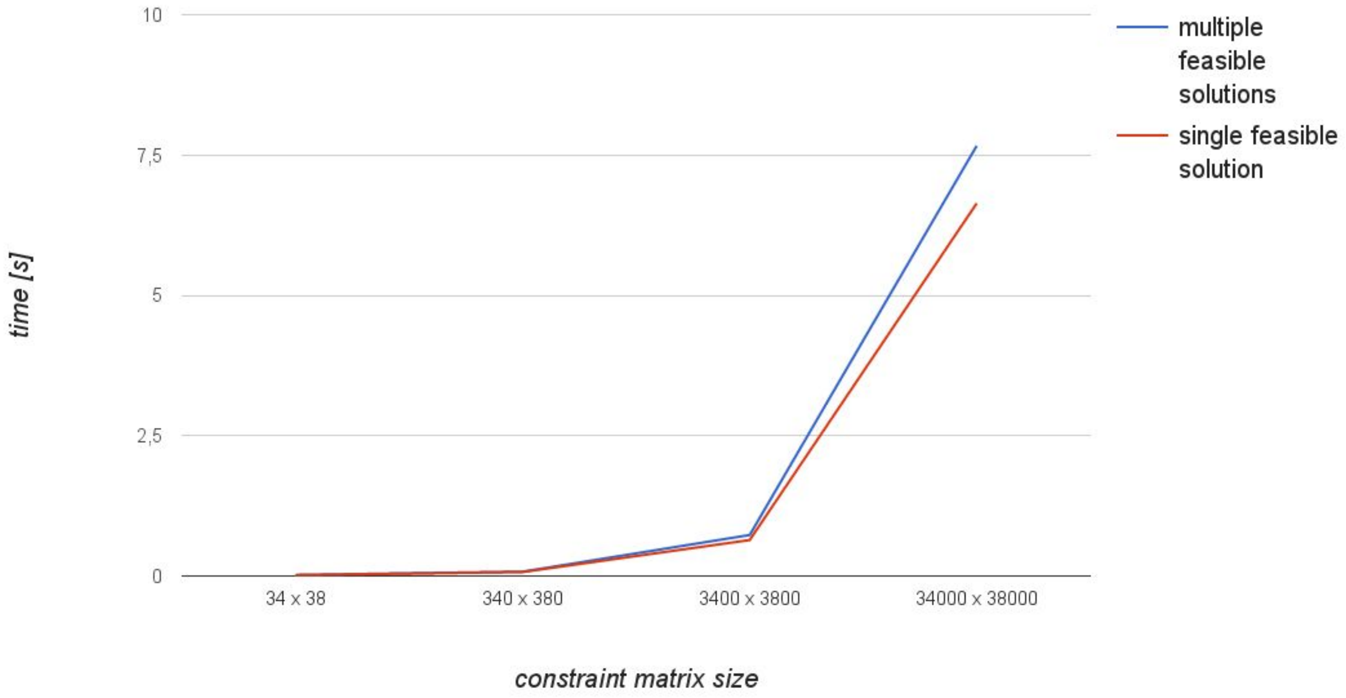
- base data size - Lucene requirements:

	no. of requirements **|R|**	available versions **|SP|**
ant	5	21
ant-junit	3	15
junit	26	2

constraint matrix size: $|A| = |R| \times |SP|$

- artificially scaled the size:
→ **n-times** more component versions and requirements; $n \in \{10, 100, 1000\}$

Simulation


10
7,5
5
2,5
0
time [s]
34 x 38
340 x 380
3400 x 3800
34000 x 38000
constraint matrix size
multiple feasible solutions
single feasible solution

Summary

- method for finding a set of components which:
    - satisfies given requirements;
    - is optimal in respect to an arbitrary metrics;
- promising performance results
- future work
    - real-data study;
    - include requirements of the searched components;
      $\longrightarrow$ create consistent transitive closure of dependencies;

Thank you for your attention.