# Learning Arbitrary RDF Dataset Enrichment Graphs Using Pre- & Postcondition Broadcasting


Kevin Dreßler

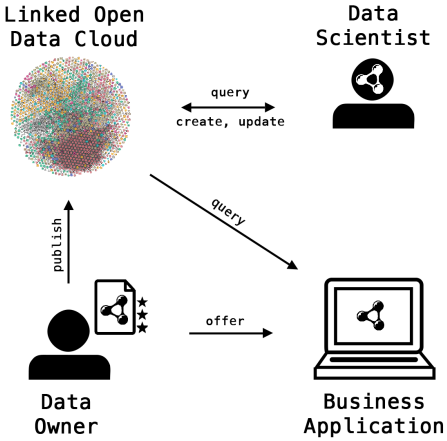
Institut für Informatik
Universität Leipzig
DICE The Data Science Group

19.07.2019

## Outline

# Motivation

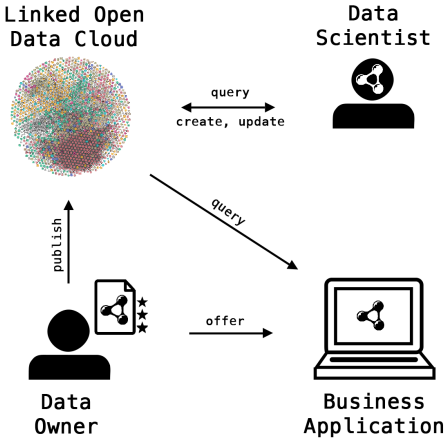
## Linked Data Integration




- Publishing 5-Star Linked Open Data (LOD)

# Motivation

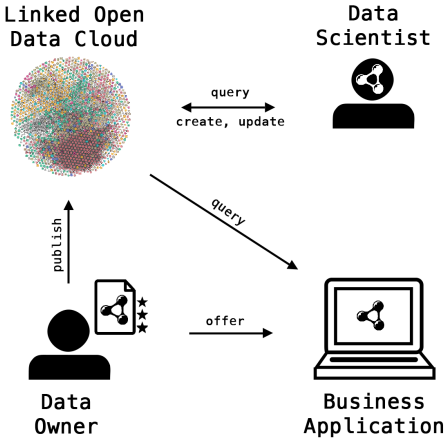
## Linked Data Integration




- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers

# Motivation

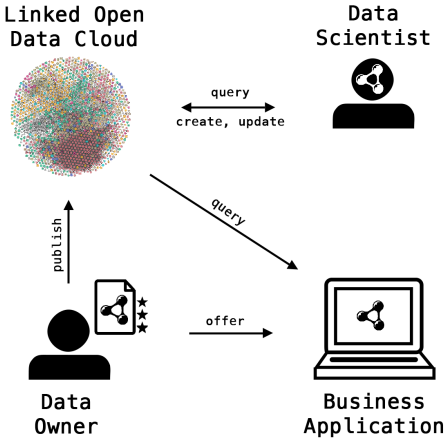
## Linked Data Integration




- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers
- Investigating research questions

# Motivation

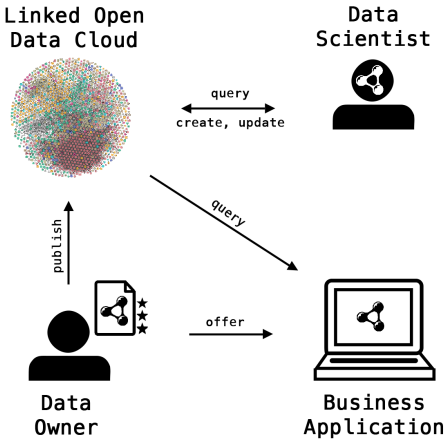
## Linked Data Integration




- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers
- Investigating research questions
- Preparing data for experiments

# Motivation

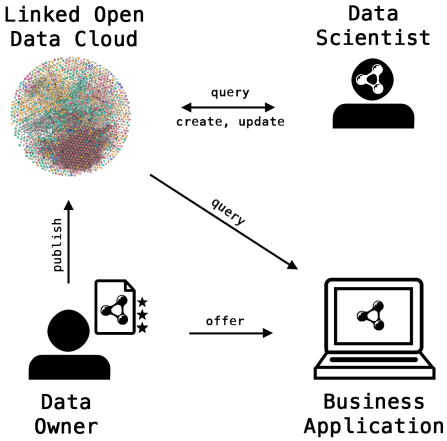
## Linked Data Integration




- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers
- Investigating research questions
- Preparing data for experiments
- Integrating LOD into business applications

# Motivation

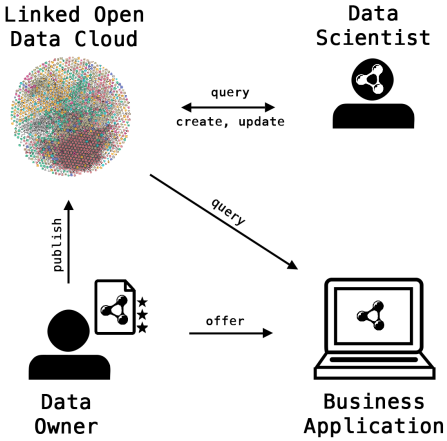
## Linked Data Integration Steps




- Lifting

# Motivation

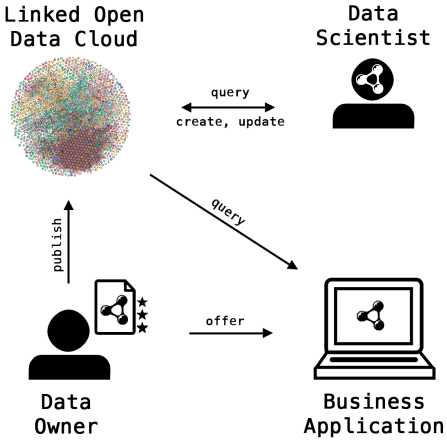
## Linked Data Integration Steps




- Lifting
- Fusion

# Motivation

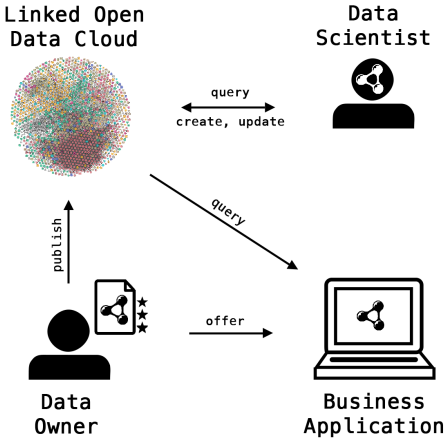
## Linked Data Integration Steps




- Lifting
- Fusion
- Linking

# Motivation

## Linked Data Integration Steps




- Lifting
- Fusion
- Linking
- **Enrichment**
  →RDF Dataset Enrichment Framework (DEER)

# Motivation

## RDF Dataset Enrichment Framework (DEER)

- Generalistic framework for RDF dataset enrichment


# Motivation

## RDF Dataset Enrichment Framework (DEER)

- Generalistic framework for RDF dataset enrichment
- Goal: accessible easy enrichment for non-experts


# Motivation

## RDF Dataset Enrichment Framework (DEER)

- Generalistic framework for RDF dataset enrichment
- Goal: accessible easy enrichment for non-experts
- Enrichment represented as pipelines


# Motivation

## RDF Dataset Enrichment Framework (DEER)

- Generalistic framework for RDF dataset enrichment
- Goal: accessible easy enrichment for non-experts
- Enrichment represented as pipelines
- Most enrichment functions need configuration


# Motivation

## RDF Dataset Enrichment Framework (DEER)

- Generalistic framework for RDF dataset enrichment
- Goal: accessible easy enrichment for non-experts
- Enrichment represented as pipelines
- Most enrichment functions need configuration
- → Requires expert knowledge ↯


# Motivation

## RDF Dataset Enrichment Framework (DEER)

- Generalistic framework for RDF dataset enrichment
- Goal: accessible easy enrichment for non-experts
- Enrichment represented as pipelines
- Most enrichment functions need configuration
- → Requires expert knowledge ↯
- → Try to use machine learning


# Motivation

## ML Algorithm in original DEER

- Training data: RDF Datasets Source ($S$) and Target ($T$)
- Fitness Function: $F_1$ score over triples
- Iterative construction with an upward refinement operator
- Good theoretical properties (finite, proper, complete, not redundant)


- But several incorrect implicit assumptions

# Motivation

## Analysis of Existing ML Algorithm in DEER

- But several incorrect implicit assumptions
  - Sequential chaining is sufficient for arbitrary enrichment ↯

- But several incorrect implicit assumptions
  - Sequential chaining is sufficient for arbitrary enrichment ↯
  - $F_1$ score over triples is a good fitness measure ↯

- But several incorrect implicit assumptions
  - Sequential chaining is sufficient for arbitrary enrichment ↯
  - $F_1$ score over triples is a good fitness measure ↯
  - Enrichment Operators are independent ↯

- But several incorrect implicit assumptions
  - Sequential chaining is sufficient for arbitrary enrichment ↯
  - $F_1$ score over triples is a good fitness measure ↯
  - Enrichment Operators are independent ↯
  - Training data always contains sufficient information for deterministic self-configuration ↯

# Motivation

## Analysis of Existing ML Algorithm in DEER

- But several incorrect implicit assumptions
  - Sequential chaining is sufficient for arbitrary enrichment ↯
  - $F_1$ score over triples is a good fitness measure ↯
  - Enrichment Operators are independent ↯
  - Training data always contains sufficient information for deterministic self-configuration ↯
- → Objective of this thesis: develop new approach (DEER2)

- Design goals: DEER2 should

- Design goals: DEER2 should
  **(G1)** be highly modular

- Design goals: DEER2 should
  - **(G1)** be highly modular
  - **(G2)** represent RDF dataset enrichment workflows efficiently as DAG

- Design goals: DEER2 should
  - **(G1)** be highly modular
  - **(G2)** represent RDF dataset enrichment workflows efficiently as DAG
  - **(G3)** include an optimized, GP based learning algorithm

- Design goals: DEER2 should
  - **(G1)** be highly modular
  - **(G2)** represent RDF dataset enrichment workflows efficiently as DAG
  - **(G3)** include an optimized, GP based learning algorithm
  - **(G4)** improve on all of the identified shortcomings of DEER.

- Design goals: DEER2 should
  - **(G1)** be highly modular
  - **(G2)** represent RDF dataset enrichment workflows efficiently as DAG
  - **(G3)** include an optimized, GP based learning algorithm
  - **(G4)** improve on all of the identified shortcomings of DEER.
- Research questions w.r.t. learning algorithm:

- Design goals: DEER2 should
  - **(G1)** be highly modular
  - **(G2)** represent RDF dataset enrichment workflows efficiently as DAG
  - **(G3)** include an optimized, GP based learning algorithm
  - **(G4)** improve on all of the identified shortcomings of DEER.
- Research questions w.r.t. learning algorithm:
  - **(Q1)** What is the optimal set of hyperparameters?

- Design goals: DEER2 should
  - **(G1)** be highly modular
  - **(G2)** represent RDF dataset enrichment workflows efficiently as DAG
  - **(G3)** include an optimized, GP based learning algorithm
  - **(G4)** improve on all of the identified shortcomings of DEER.
- Research questions w.r.t. learning algorithm:
  - **(Q1)** What is the optimal set of hyperparameters?
  - **(Q2)** How does our approach perform on real world datasets?


## Definition **Dataset Operator**

$$\mathbb{O}_{(n,m)} : \mathbb{D}^n \times \mathbb{D} \to \mathbb{D}^m$$
$$\left(\left(D_1^{(\mathsf{in})}, \ldots, D_n^{(\mathsf{in})}\right), P\right) \mapsto \left(D_1^{(\mathsf{out})}, \ldots, D_m^{(\mathsf{out})}\right)$$

- $\mathbb{O}_{(0,1)}$ is called a **dataset emitter**,


## Definition **Dataset Operator**

$$\mathbb{O}_{(n,m)} : \mathbb{D}^n \times \mathbb{D} \to \mathbb{D}^m$$
$$\left(\left(D_1^{(\text{in})}, \ldots, D_n^{(\text{in})}\right), P\right) \mapsto \left(D_1^{(\text{out})}, \ldots, D_m^{(\text{out})}\right)$$

- $\mathbb{O}_{(0,1)}$ is called a **dataset emitter**,
- $\mathbb{O}_{(1,0)}$ is called a **dataset acceptor**,


## Definition **Dataset Operator**

$$\mathbb{O}_{(n,m)} : \mathbb{D}^n \times \mathbb{D} \to \mathbb{D}^m$$
$$\left(\left(D_1^{(\text{in})}, \ldots, D_n^{(\text{in})}\right), P\right) \mapsto \left(D_1^{(\text{out})}, \ldots, D_m^{(\text{out})}\right)$$

- $\mathbb{O}_{(0,1)}$ is called a **dataset emitter**,
- $\mathbb{O}_{(1,0)}$ is called a **dataset acceptor**,
- $\mathbb{O}_{(n,m)}$ is called an **enrichment operator** for $n, m > 0$ and

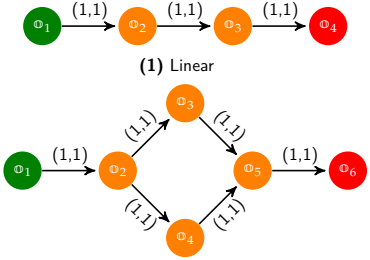

## Definition **Dataset Operator**

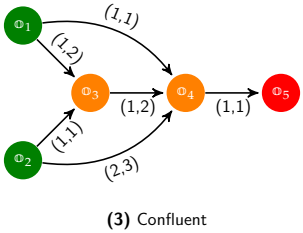
$$\mathbb{O}_{(n,m)} : \mathbb{D}^n \times \mathbb{D} \to \mathbb{D}^m$$
$$\left(\left(D_1^{(\text{in})}, \ldots, D_n^{(\text{in})}\right), P\right) \mapsto \left(D_1^{(\text{out})}, \ldots, D_m^{(\text{out})}\right)$$

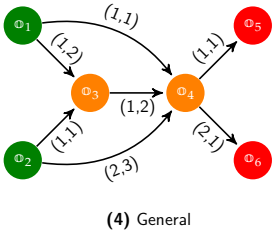
- $\mathbb{O}_{(0,1)}$ is called a **dataset emitter**,
- $\mathbb{O}_{(1,0)}$ is called a **dataset acceptor**,
- $\mathbb{O}_{(n,m)}$ is called an **enrichment operator** for $n, m > 0$ and
- $\mathbb{O}_{(n,1)}$ is called a **confluent enrichment operator**.

# Approach

## Definitions (Cont.)



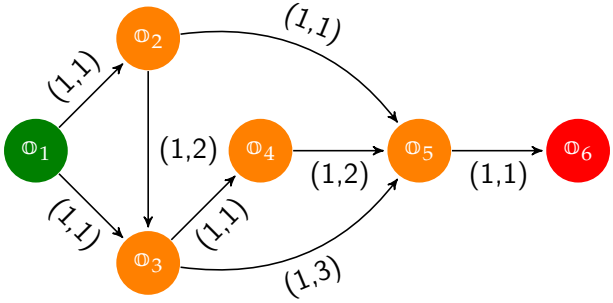




## Definition **Enrichment Graph**

Let $G = (\mathbf{V}, \mathbf{E}, \mathbf{L}, \Phi, \Psi)$ be a directed acyclic labeled multigraph.

$$\mathbf{L}\colon E \to 2^{(\mathbb{N}\times\mathbb{N})} \qquad \Phi\colon \mathbf{V} \to \mathbb{O} \qquad \Psi\colon \mathbf{V} \to \mathcal{D}$$

$$e = (u, v) \mapsto \{(i_1, j_1), \ldots, (i_n, j_n)\} \qquad v \mapsto \mathbb{o}_{(n,m)} \qquad v \mapsto P$$




| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathbb{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathbb{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathbb{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathbb{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 5: | $\mathbb{O}_5$ | $P_5$ | 3 | 2 | 4 | 3 |

- Restrict leaning problem to

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$
  - regularly shaped RDF datasets

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$
  - regularly shaped RDF datasets
- Training data

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$
  - regularly shaped RDF datasets
- Training data
  - source training datasets(s) ($\leq 2$)

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$
  - regularly shaped RDF datasets
- Training data
  - source training datasets(s) ($\leq 2$)
  - and target training dataset ($= 1$)

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$
  - regularly shaped RDF datasets
- Training data
  - source training datasets(s) ($\leq 2$)
  - and target training dataset ($= 1$)
- Learn enrichment table using Genetic Programming (GP) and Multi-Expression Programming (MEP)

- Restrict leaning problem to
  - **inherently confluent enrichment graphs**
  - enrichment operators with in-degree in $\{1, 2\}$
  - regularly shaped RDF datasets
- Training data
  - source training datasets(s) ($\leq 2$)
  - and target training dataset ($= 1$)
- Learn enrichment table using Genetic Programming (GP) and Multi-Expression Programming (MEP)
- Use linear combination of $F_1$ measure on subjects, predicates, objects and whole triples as fitness function

- $(\mu + \lambda)$ Genetic Algorithm:

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$
  - Fill the mating pool $p_m$ with $\mu$ individuals using tournament selection on $p_c$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$
  - Fill the mating pool $p_m$ with $\mu$ individuals using tournament selection on $p_c$
  - Apply crossover randomly to pairs $(\mathbb{T}_1, \mathbb{T}_2) \in p_m$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$
  - Fill the mating pool $p_m$ with $\mu$ individuals using tournament selection on $p_c$
  - Apply crossover randomly to pairs $(\mathbb{T}_1, \mathbb{T}_2) \in p_m$
  - Insert resulting childs into $p_n$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$
  - Fill the mating pool $p_m$ with $\mu$ individuals using tournament selection on $p_c$
  - Apply crossover randomly to pairs $(\mathbb{T}_1, \mathbb{T}_2) \in p_m$
  - Insert resulting childs into $p_n$
  - Insert $\lambda - \mu$ individuals into $p_n$ using tournament selection on $p_c$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$
  - Fill the mating pool $p_m$ with $\mu$ individuals using tournament selection on $p_c$
  - Apply crossover randomly to pairs $(\mathbb{T}_1, \mathbb{T}_2) \in p_m$
  - Insert resulting childs into $p_n$
  - Insert $\lambda - \mu$ individuals into $p_n$ using tournament selection on $p_c$
  - Apply mutation to each individual $\mathbb{T} \in p_n$ with probability $\sigma$

- $(\mu + \lambda)$ Genetic Algorithm:
- Initialize current population $p_c$
- Repeat until termination criterion is met:
  - Evaluate the current population $p_c$
  - Initialize the next population with elite $p_n := \{\text{best}(p_c)\}$
  - Fill the mating pool $p_m$ with $\mu$ individuals using tournament selection on $p_c$
  - Apply crossover randomly to pairs $(\mathbb{T}_1, \mathbb{T}_2) \in p_m$
  - Insert resulting childs into $p_n$
  - Insert $\lambda - \mu$ individuals into $p_n$ using tournament selection on $p_c$
  - Apply mutation to each individual $\mathbb{T} \in p_n$ with probability $\sigma$
  - For each row $\mathbb{T}_i$ of individuals chosen for mutation, mutate the row with probability $\rho$

- MEP: One genotype represents multiple solutions (phenotypes)

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathfrak{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathfrak{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathfrak{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathfrak{O}_6$ | $P_6$ | 2 | 3 | 1 | 0 |
| 5: | $\mathfrak{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 6: | $\mathfrak{O}_7$ | $P_7$ | 1 | 5 | 0 | 0 |
| 7: | $\mathfrak{O}_8$ | $P_8$ | 2 | 4 | 3 | 0 |
| 8: | $\mathfrak{O}_5$ | $P_5$ | 3 | 2 | 5 | 3 |
| 9: | $\mathfrak{O}_9$ | $P_9$ | 1 | 7 | 0 | 0 |

- MEP: One genotype represents multiple solutions (phenotypes)
- Use MEP for enrichment tables

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathfrak{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathfrak{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathfrak{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathfrak{O}_6$ | $P_6$ | 2 | 3 | 1 | 0 |
| 5: | $\mathfrak{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 6: | $\mathfrak{O}_7$ | $P_7$ | 1 | 5 | 0 | 0 |
| 7: | $\mathfrak{O}_8$ | $P_8$ | 2 | 4 | 3 | 0 |
| 8: | $\mathfrak{O}_5$ | $P_5$ | 3 | 2 | 5 | 3 |
| 9: | $\mathfrak{O}_9$ | $P_9$ | 1 | 7 | 0 | 0 |

- MEP: One genotype represents multiple solutions (phenotypes)
- Use MEP for enrichment tables
- Called multi-expressive enrichment table

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathfrak{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathfrak{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathfrak{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathfrak{O}_6$ | $P_6$ | 2 | 3 | 1 | 0 |
| 5: | $\mathfrak{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 6: | $\mathfrak{O}_7$ | $P_7$ | 1 | 5 | 0 | 0 |
| 7: | $\mathfrak{O}_8$ | $P_8$ | 2 | 4 | 3 | 0 |
| 8: | $\mathfrak{O}_5$ | $P_5$ | 3 | 2 | 5 | 3 |
| 9: | $\mathfrak{O}_9$ | $P_9$ | 1 | 7 | 0 | 0 |

- Add all dataset emitters

- Add all dataset emitters
- Randomly generate rows

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathfrak{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathfrak{O}_2$ | $\emptyset$ | 1 | 1 | 0 | 0 |

- Add all dataset emitters
- Randomly generate rows

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathfrak{o}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathfrak{o}_2$ | $\emptyset$ | 1 | 1 | 0 | 0 |
| 3: | $\mathfrak{o}_3$ | $\emptyset$ | 2 | 1 | 2 | 0 |
| 4: | $\mathfrak{o}_6$ | $\emptyset$ | 2 | 3 | 1 | 0 |
| 5: | $\mathfrak{o}_4$ | $\emptyset$ | 1 | 3 | 0 | 0 |
| 6: | $\mathfrak{o}_7$ | $\emptyset$ | 1 | 5 | 0 | 0 |
| 7: | $\mathfrak{o}_8$ | $\emptyset$ | 2 | 4 | 3 | 0 |
| 8: | $\mathfrak{o}_5$ | $\emptyset$ | 3 | 2 | 5 | 3 |
| 9: | $\mathfrak{o}_9$ | $\emptyset$ | 1 | 7 | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathbb{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathbb{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathbb{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathbb{O}_6$ | $P_6$ | 2 | 3 | 1 | 0 |
| 5: | $\mathbb{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 6: | $\mathbb{O}_7$ | $P_7$ | 1 | 5 | 0 | 0 |
| 7: | $\mathbb{O}_8$ | $P_8$ | 2 | 4 | 3 | 0 |
| 8: | $\mathbb{O}_5$ | $P_5$ | 3 | 2 | 5 | 3 |
| 9: | $\mathbb{O}_9$ | $P_9$ | 1 | 7 | 0 | 0 |

→

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathbb{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathbb{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathbb{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathbb{O}_6$ | $P_6$ | 2 | 3 | 1 | 0 |
| 5: | $\mathbb{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 6: | $\mathbb{O}_7$ | $P_7$ | 1 | 5 | 0 | 0 |
| 7: | $\mathbb{O}_8$ | $P_8$ | 2 | 4 | 3 | 0 |
| 8: | $\mathbb{O}_5$ | $P_5$ | 3 | 2 | 5 | 3 |
| 9: | $\mathbb{O}_9$ | $P_9$ | 1 | 7 | 0 | 0 |

# Approach

## Enrichment Table Compaction II

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathbb{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathbb{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathbb{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathbb{O}_6$ | $P_6$ | 2 | 3 | 1 | 0 |
| 5: | $\mathbb{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 6: | $\mathbb{O}_7$ | $P_7$ | 1 | 5 | 0 | 0 |
| 7: | $\mathbb{O}_8$ | $P_8$ | 2 | 5 | 3 | 0 |
| 8: | $\mathbb{O}_5$ | $P_5$ | 3 | 2 | 5 | 3 |
| 9: | $\mathbb{O}_9$ | $P_9$ | 1 | 7 | 0 | 0 |

→

| | | | | | | |
|---|---|---|---|---|---|---|
| 1: | $\mathbb{O}_1$ | $P_1$ | 0 | 0 | 0 | 0 |
| 2: | $\mathbb{O}_2$ | $P_2$ | 1 | 1 | 0 | 0 |
| 3: | $\mathbb{O}_3$ | $P_3$ | 2 | 1 | 2 | 0 |
| 4: | $\mathbb{O}_4$ | $P_4$ | 1 | 3 | 0 | 0 |
| 5: | $\mathbb{O}_5$ | $P_5$ | 3 | 2 | 4 | 3 |

- Take into account the problem domain

- Take into account the problem domain
- More like a guided search, not completely random

- Take into account the problem domain
- More like a guided search, not completely random
- Intereseting properties of our problem domain:

- Take into account the problem domain
- More like a guided search, not completely random
- Intereseting properties of our problem domain:
  - Structure of the graph

- Take into account the problem domain
- More like a guided search, not completely random
- Intereseting properties of our problem domain:
  - Structure of the graph
  - Applicability of enrichment operators

- Idea: select two most promising phenotypes

- Idea: select two most promising phenotypes
- → combine them into a new genotype

- Idea: select two most promising phenotypes
- → combine them into a new genotype
- With probability of 0.25: insert an enrichment operator merging both together

- Idea: method on enrichment operators states applicability

- Idea: method on enrichment operators states applicability
- Select row for mutation

- Idea: method on enrichment operators states applicability
- Select row for mutation
- Broadcast pre-/postconditions to operators

- Idea: method on enrichment operators states applicability
- Select row for mutation
- Broadcast pre-/postconditions to operators
- Operators return applicability

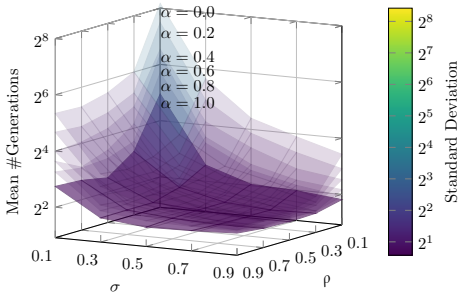
- Idea: method on enrichment operators states applicability
- Select row for mutation
- Broadcast pre-/postconditions to operators
- Operators return applicability
- Roulette wheel selection proportionate on applicability

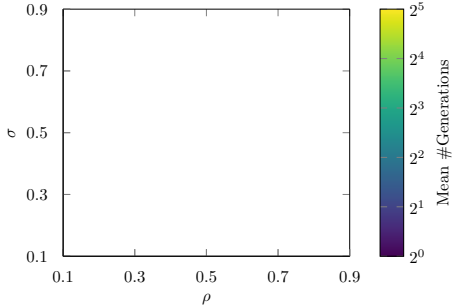
- **Fist set of experiments:** Hyperparameter Optimization
  - Grid search
  - Mutation probability $\sigma \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$
  - Mutation rate $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$
  - Offspring fraction $\alpha = \frac{\lambda}{\mu} \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
- **Second set of experiments:** Performance on real world example

# Evaluation

## Hyperparameter Optimization

- Very simple toy learning problem
- 3 enrichment operators, order does not matter
- only baseline algorithm $(\mu = 30, r = 7, g = 5000)$
- 1000 repetitions
- best set of parameter $(\alpha/\sigma/\rho) = (1.0/0.9/0.9)$

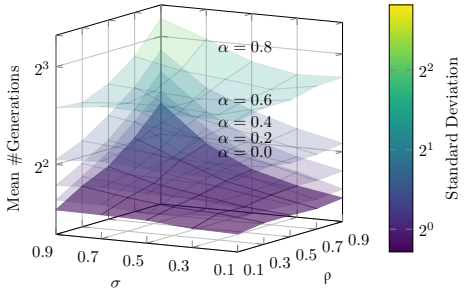


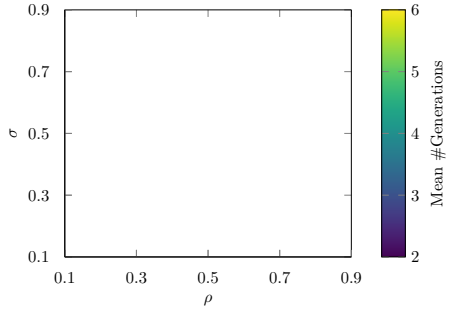



# Evaluation

## Hyperparameter Optimization

- Very simple toy learning problem
- 3 enrichment operators, order does not matter
- with enrichment table compaction enabled
- 1000 repetitions
- best set of parameter $(\alpha/\sigma/\rho) = (0.0/0.1/0.1)$

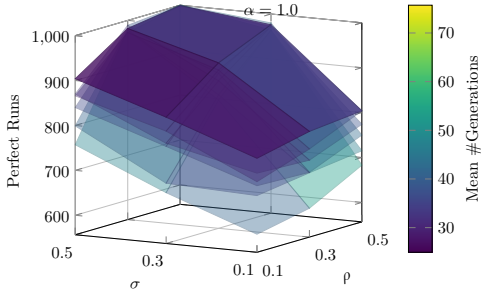


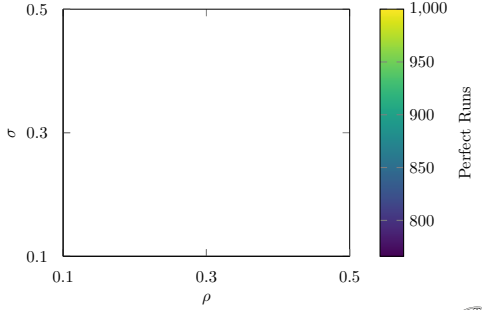



# Evaluation

## Hyperparameter Optimization

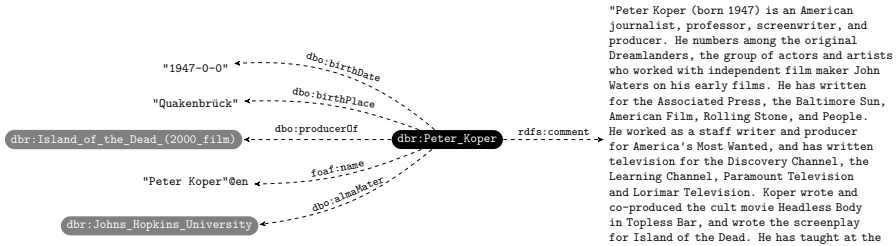
- Much harder problem
- 5 enrichment operators, order *does matter*
- with all optimizations enabled
- 1000 repetitions
- best set of parameter $(\alpha/\sigma/\rho) = (1.0/0.5/0.5)$



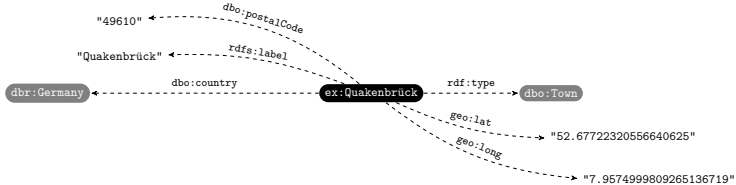



# Evaluation

## Performance on Real World Example



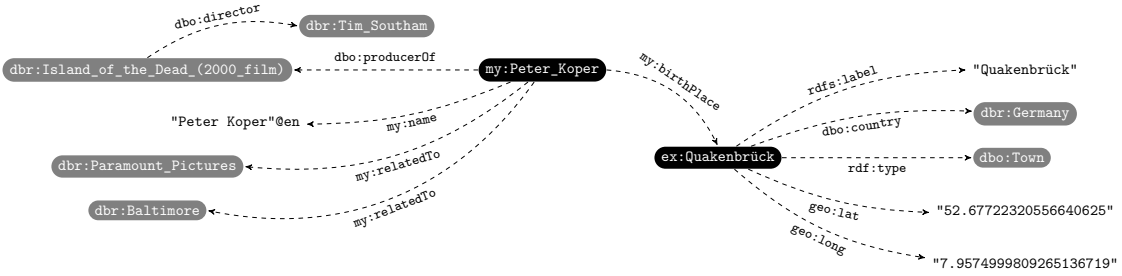

**(1)**



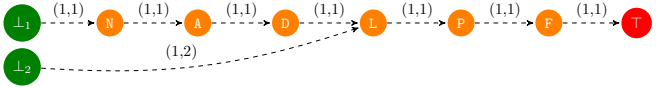

**(2)**

# Evaluation

## Performance on Real World Example



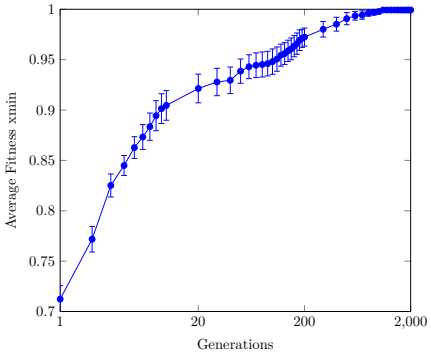

**(3)**




**(4)**

# Evaluation

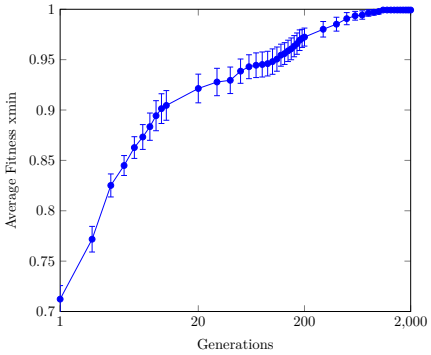
## Performance on Real World Example




- 50 repetitions

# Evaluation

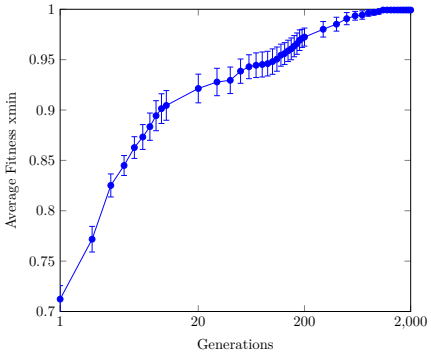
## Performance on Real World Example




- maximum 2000 generations
- 50 repetitions

# Evaluation

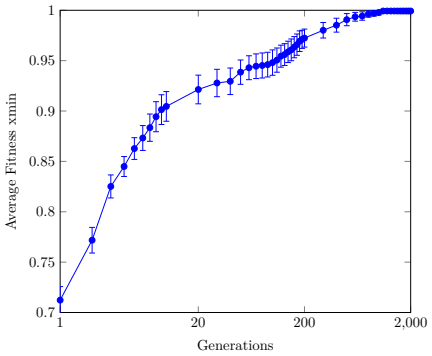
## Performance on Real World Example




- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- 50 repetitions

# Evaluation

## Performance on Real World Example

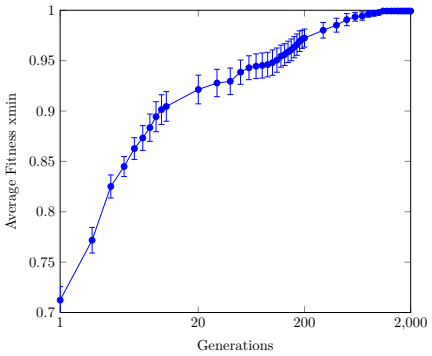



- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- measure best fitness in each generation

- 50 repetitions

# Evaluation

## Performance on Real World Example

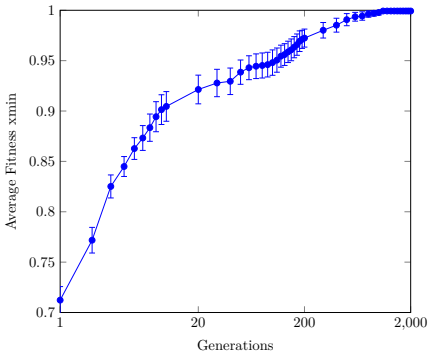



- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- measure best fitness in each generation
- compute average fitness for each generation

- 50 repetitions

# Evaluation

## Performance on Real World Example

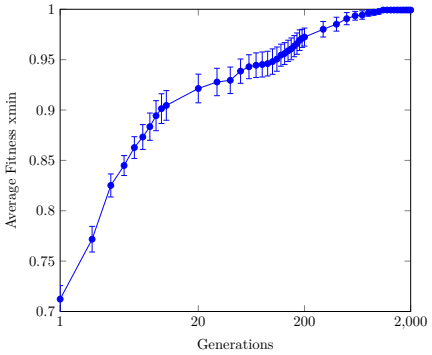



- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- measure best fitness in each generation
- compute average fitness for each generation
- compute 95%-confidence intervals using Students $t$-distribution

- 50 repetitions

# Evaluation

## Performance on Real World Example




- 50 repetitions

- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- measure best fitness in each generation
- compute average fitness for each generation
- compute 95%-confidence intervals using Students $t$-distribution
- Result: on average solution quality over 99% after 500 generations with 95%-confidence interval of $\pm 0.6\%$

# Conclusion

- Implemented a tool for generalistic DAG-shaped enrichment workflows

# Conclusion

- Implemented a tool for generalistic DAG-shaped enrichment workflows
- Developed a classification of enrichment graphs

# Conclusion

- Implemented a tool for generalistic DAG-shaped enrichment workflows
- Developed a classification of enrichment graphs
- Derived a GP- and MEP-based learning approach for inherently confluent enrichment graphs

# Conclusion

- Implemented a tool for generalistic DAG-shaped enrichment workflows
- Developed a classification of enrichment graphs
- Derived a GP- and MEP-based learning approach for inherently confluent enrichment graphs
- Supplied several optimizations

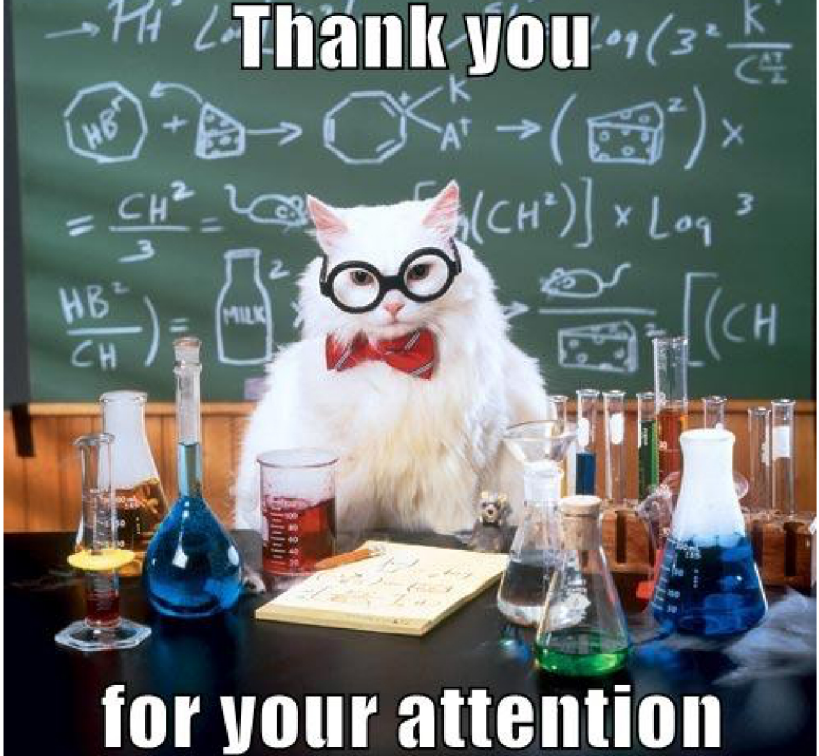
# Conclusion

- Implemented a tool for generalistic DAG-shaped enrichment workflows
- Developed a classification of enrichment graphs
- Derived a GP- and MEP-based learning approach for inherently confluent enrichment graphs
- Supplied several optimizations
- Optimizations successfully improved solution quality & time to convergence

# Conclusion

- Implemented a tool for generalistic DAG-shaped enrichment workflows
- Developed a classification of enrichment graphs
- Derived a GP- and MEP-based learning approach for inherently confluent enrichment graphs
- Supplied several optimizations
- Optimizations successfully improved solution quality & time to convergence
- Over $99\%$ after $500$ generations with $95\%$-confidence interval of $\pm 0.6\%$ in a real-world test case

# Thank you
# for your Attention!


Thank you
for your attention