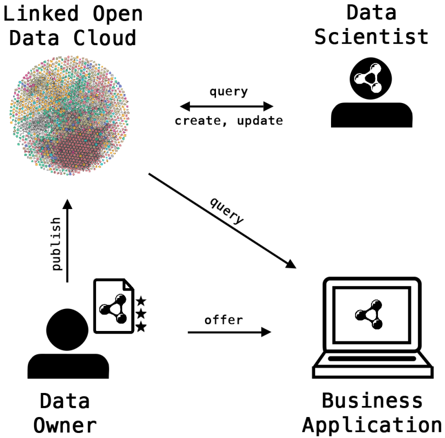
Learning Arbitrary RDF Dataset Enrichment Graphs Using Pre- & Postcondition Broadcasting

Kevin Dreßler

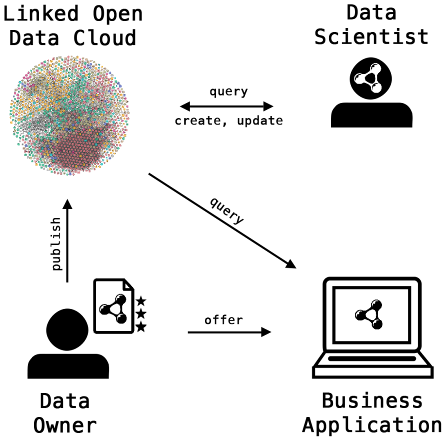
Institut für Informatik
Universität Leipzig
DICE The Data Science Group

19.07.2019

Outline



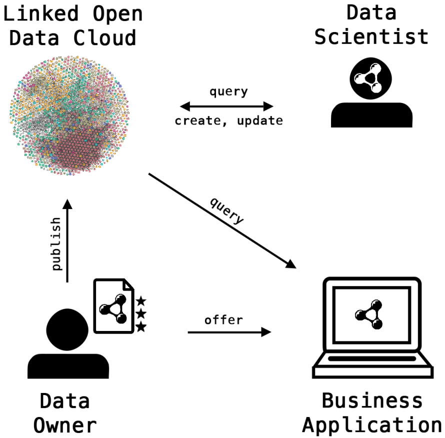
- Publishing 5-Star Linked Open Data (LOD)



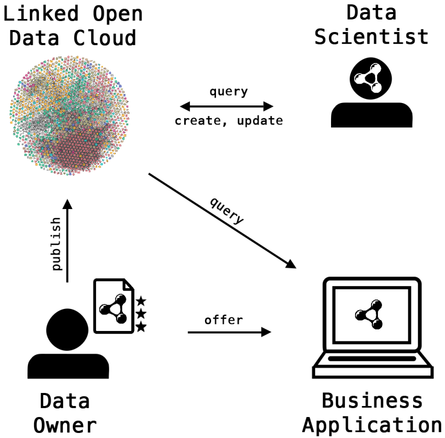
- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers

Motivation

Linked Data Integration



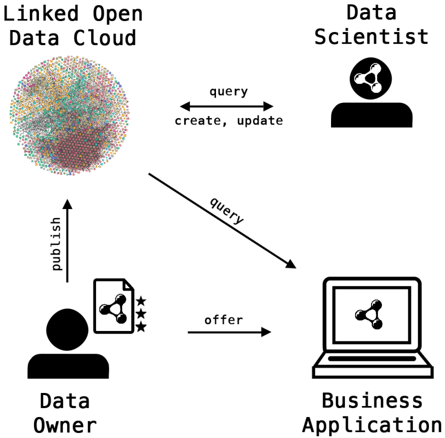
- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers
- Investigating research questions



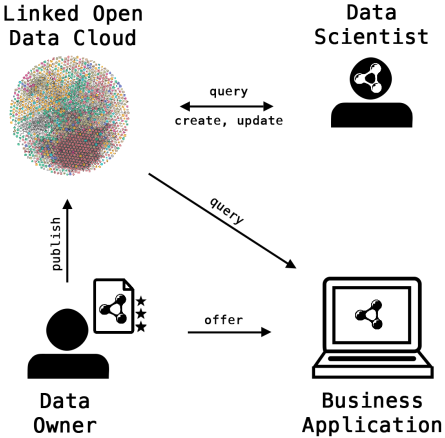
- Publishing 5-Star Linked Open Data (LOD)
- Flexible distribution of datasets to customers
- Investigating research questions
- Preparing data for experiments

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

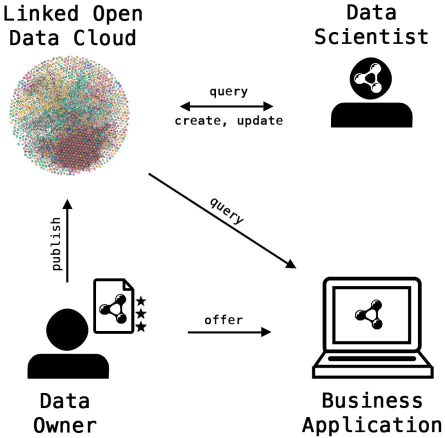


Linked Open Data Cloud
Data Scientist
query
create, update
query
publish
offer
Data Owner
Business Application

- Lifting

Motivation

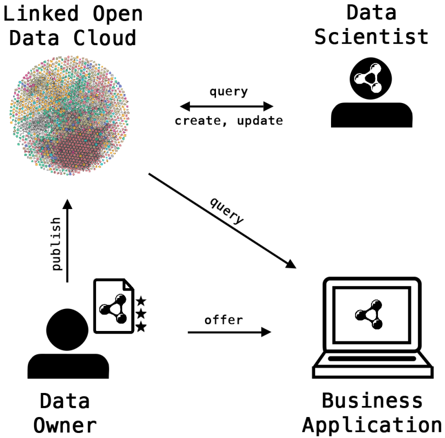
Linked Data Integration Steps



- Lifting
- Fusion

Motivation

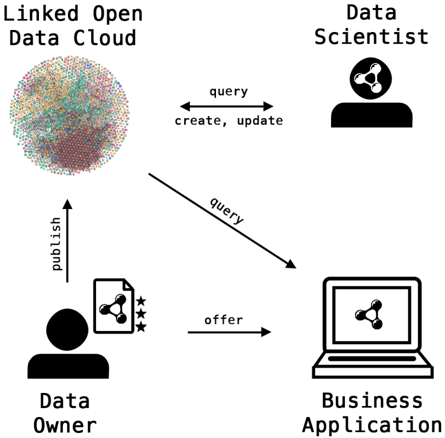
Linked Data Integration Steps



- 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 $\lightning$

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

Motivation

Analysis of Existing ML Algorithm in DEER

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

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 ↯

Motivation

Analysis of Existing ML Algorithm in DEER

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

- Design goals: DEER2 should

Motivation

Derived Goals & Research Questions

- Design goals: DEER2 should

(G1) be highly modular

Motivation

Derived Goals & Research Questions

- 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:

Motivation

Derived Goals & Research Questions

- 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?

Motivation

Derived Goals & Research Questions

- 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?

$$\mathbb{O}_{(0,1)} \longrightarrow \mathbb{O}_{(1,1)} \longrightarrow \mathbb{O}_{(1,0)}$$

Definition **Dataset Operator**

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

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

$$\mathbb{O}_{(0,1)} \longrightarrow \mathbb{O}_{(1,1)} \longrightarrow \mathbb{O}_{(1,0)}$$

Definition **Dataset Operator**

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

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

$$\mathbb{O}_{(0,1)} \longrightarrow \mathbb{O}_{(1,1)} \longrightarrow \mathbb{O}_{(1,0)}$$

Definition **Dataset Operator**

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

- $\mathbb{O}_{(0,1)}$ 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

Approach

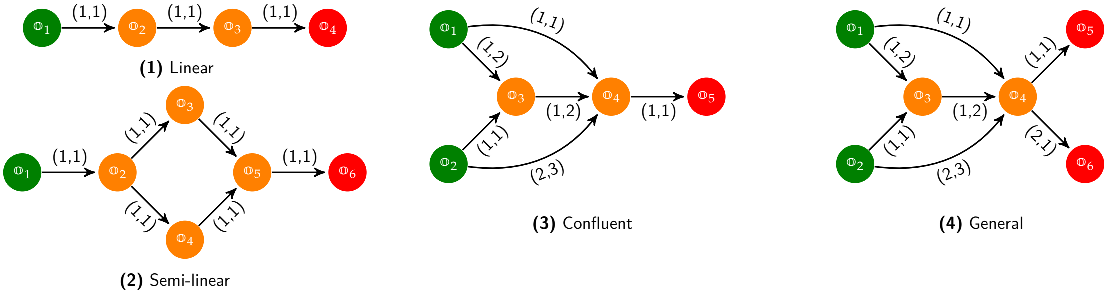
Definitions

$$\mathbb{O}_{(0,1)} \longrightarrow \mathbb{O}_{(1,1)} \longrightarrow \mathbb{O}_{(1,0)}$$

Definition **Dataset Operator**

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

- $\mathbb{O}_{(0,1)}$ 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**.



Definition **Enrichment Graph**

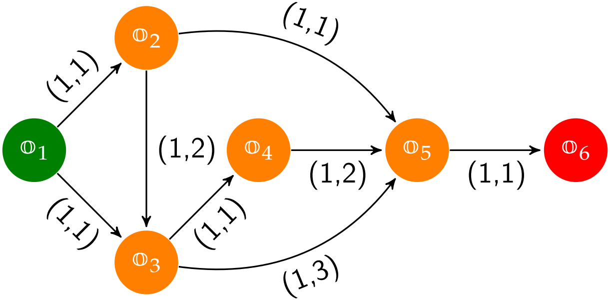
Let $G = (\mathbf{V}, \mathbf{E}, \mathbf{L}, \Phi, \Psi)$ be a directed acyclic labeled multigraph.

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

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

Approach

Enrichment Tables - An Efficient Representation for Learning



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

Approach
Learning Problem

- 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

Approach

Learning Problem

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

Approach

Learning Problem

- 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) (≤ 2)

Approach

Learning Problem

- 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) (≤ 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) (≤ 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) (≤ 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

Approach

Genetic Programming Algorithm

- $(\mu + \lambda)$ 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 $p_n := \{\text{best}(p_c)\}$

Approach

Genetic Programming Algorithm

- $(\mu + \lambda)$ 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 μ 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 μ individuals using tournament selection on p_c
  - Apply crossover randomly to pairs $(\mathbb{T}_1, \mathbb{T}_2) \in p_m$

Approach

Genetic Programming Algorithm

- $(\mu + \lambda)$ 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 μ 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 μ 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 μ 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 σ

Approach

Genetic Programming Algorithm

- $(\mu + \lambda)$ 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 μ 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 σ
  - For each row $\mathbb{T}_i$ of individuals chosen for mutation, mutate the row with probability ρ

- MEP: One genotype represents multiple solutions (phenotypes)

1:	ϕ_1	P_1	0	0	0	0
2:	ϕ_2	P_2	1	1	0	0
3:	ϕ_3	P_3	2	1	2	0
4:	ϕ_6	P_6	2	3	1	0
5:	ϕ_4	P_4	1	3	0	0
6:	ϕ_7	P_7	1	5	0	0
7:	ϕ_8	P_8	2	4	3	0
8:	ϕ_5	P_5	3	2	5	3
9:	ϕ_9	P_9	1	7	0	0

Approach

Multi-Expressive Enrichment Tables

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

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

Multi-Expressive Enrichment Tables

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

1:	ω_1	P_1	0	0	0	0
2:	ω_2	P_2	1	1	0	0
3:	ω_3	P_3	2	1	2	0
4:	ω_6	P_6	2	3	1	0
5:	ω_4	P_4	1	3	0	0
6:	ω_7	P_7	1	5	0	0
7:	ω_8	P_8	2	4	3	0
8:	ω_5	P_5	3	2	5	3
9:	ω_9	P_9	1	7	0	0

Approach

Population Initialization

- Add all dataset emitters

1: $\mathbb{o}_1$	P_1	0	0	0	0

Approach

Population Initialization

- Add all dataset emitters
- Randomly generate rows

1:	$\mathbb{O}_1$	P_1	0	0	0	0
2:	$\mathbb{O}_2$	$\emptyset$	1	1	0	0

Approach

Population Initialization

- Add all dataset emitters
- Randomly generate rows

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

Approach

Enrichment Table Compaction

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

$\rightarrow$

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:	ω_1	P_1	0	0	0	0
2:	ω_2	P_2	1	1	0	0
3:	ω_3	P_3	2	1	2	0
4:	ω_6	P_6	2	3	1	0
5:	ω_4	P_4	1	3	0	0
6:	ω_7	P_7	1	5	0	0
7:	ω_8	P_8	2	5	3	0
8:	ω_5	P_5	3	2	5	3
9:	ω_9	P_9	1	7	0	0

$\rightarrow$

1:	ω_1	P_1	0	0	0	0
2:	ω_2	P_2	1	1	0	0
3:	ω_3	P_3	2	1	2	0
4:	ω_4	P_4	1	3	0	0
5:	ω_5	P_5	3	2	**4**	3

Approach

Semantic Genetic Operators

- Take into account the problem domain

Approach

Semantic Genetic Operators

- 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

Approach

Semantic Genetic Operators

- 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

Approach

Pre- & Postcondition Mutation

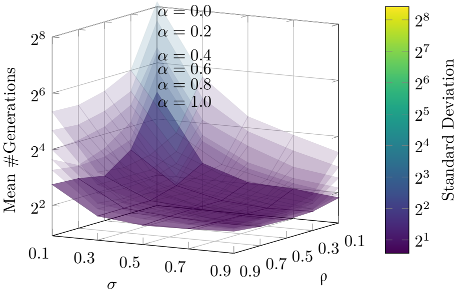
- Idea: method on enrichment operators states applicability
- Select row for mutation
- Broadcast pre-/postconditions to operators
- Operators return applicability
- Roulette wheel selection proportionate on applicability

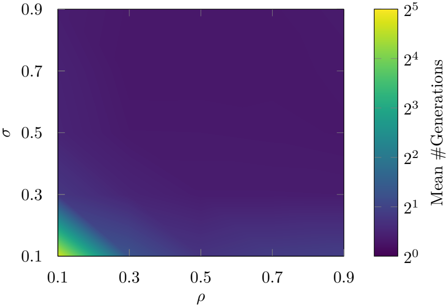
- **Fist set of experiments:** Hyperparameter Optimization
  - Grid search
  - Mutation probability $\sigma \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$
  - Mutation rate $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$
  - Offspring fraction $\alpha = \frac{\lambda}{\mu} \in \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$
- **Second set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

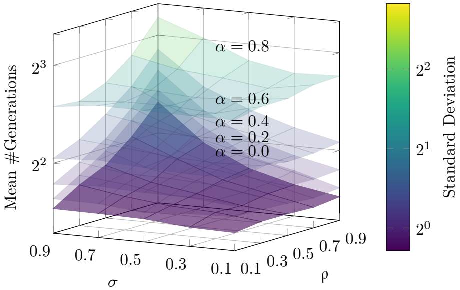


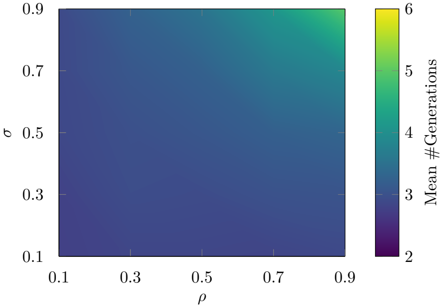


Evaluation

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 $(\alpha/\sigma/\rho) = (0.0/0.1/0.1)$

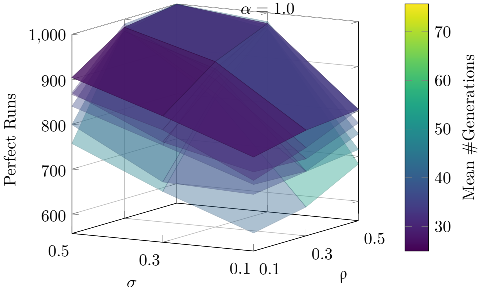


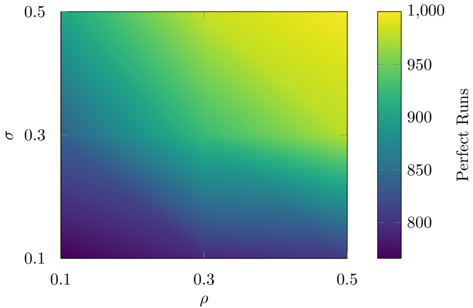


Evaluation

Hyperparameter Optimization

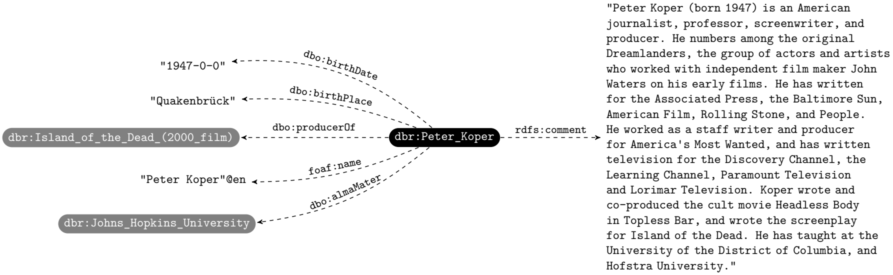
- Much harder problem
- 5 enrichment operators, order *does matter*
- with all optimizations enabled
- 1000 repetitions
- best set of parameter $(\alpha/\sigma/\rho) = (1.0/0.5/0.5)$



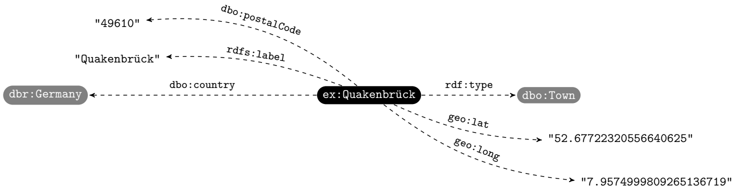


Evaluation

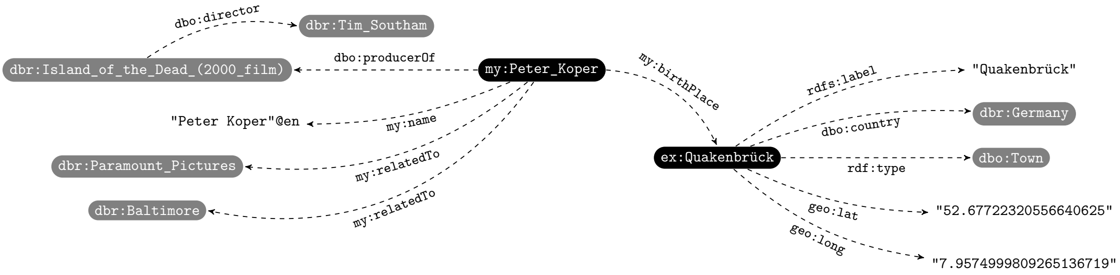
Performance on Real World Example



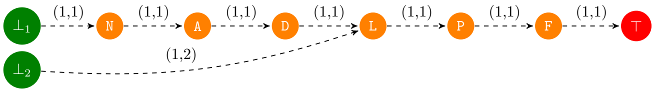
(1)



(2)


dbo:director
dbr:Tim_Southam
dbr:Island_of_the_Dead_(2000_film)
dbo:producerOf
my:Peter_Koper
my:birthPlace
"Peter Koper"@en
my:name
dbr:Paramount_Pictures
my:relatedTo
dbr:Baltimore
my:relatedTo
rdfs:label
"Quakenbrück"
dbr:Germany
dbo:country
ex:Quakenbrück
dbo:Town
rdf:type
geo:lat
"52.67722320556640625"
geo:long
"7.95749998092651367 19"

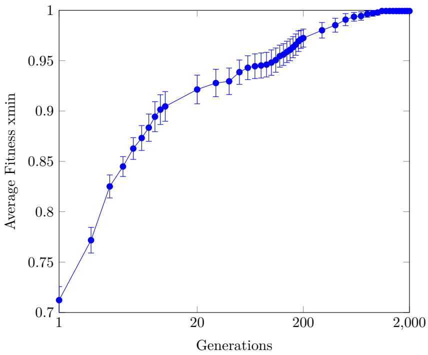
(3)


⊥1
(1,1)
N
(1,1)
A
(1,1)
D
(1,1)
L
(1,1)
P
(1,1)
F
(1,1)
⊤
⊥2
(1,2)

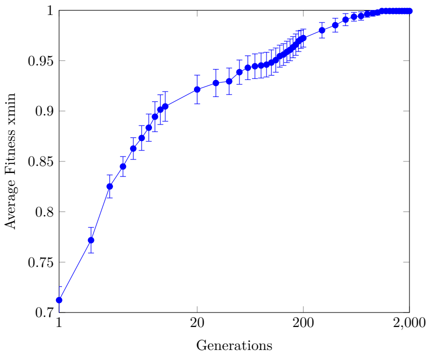
(4)

Evaluation

Performance on Real World Example



- 50 repetitions

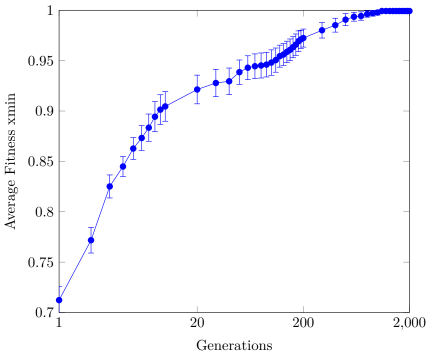


- maximum 2000 generations

- 50 repetitions

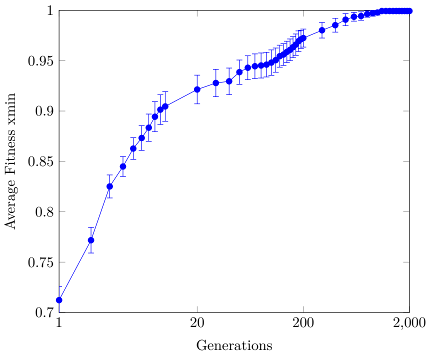
Evaluation

Performance on Real World Example

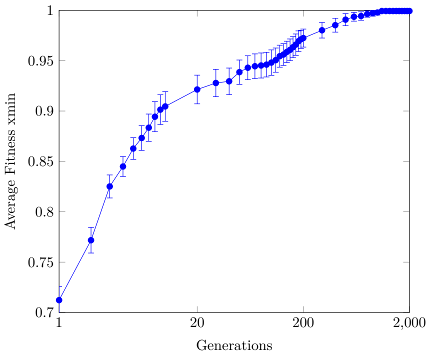


- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$

- 50 repetitions



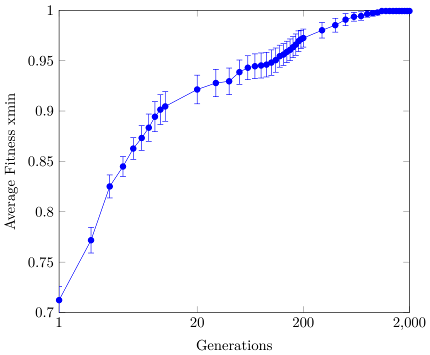
- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- measure best fitness in each generation
- 50 repetitions



- 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
- 50 repetitions

Evaluation

Performance on Real World Example

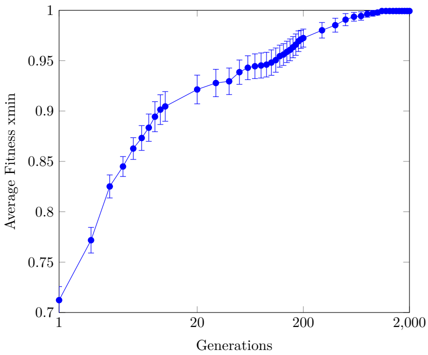


- maximum 2000 generations
- $\mu = 30, \alpha = 1, \sigma = 0.5, \rho = 0.5$
- measure best fitness in each generation
- compute average fitness for each generation
- compute 95%-confidence intervals using Students t-distribution

- 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

Conclusion

Thank You for Your Attention