

**S+SSPR
2018**

THE VF3-LIGHT SUBGRAPH ISOMORPHISM ALGORITHM: WHEN DOING LESS IS MORE EFFECTIVE

V. Carletti, P. Foggia, A. Saggese, M. Vento
University of Salerno



UNIVERSITÀ DEGLI STUDI DI SALERNO



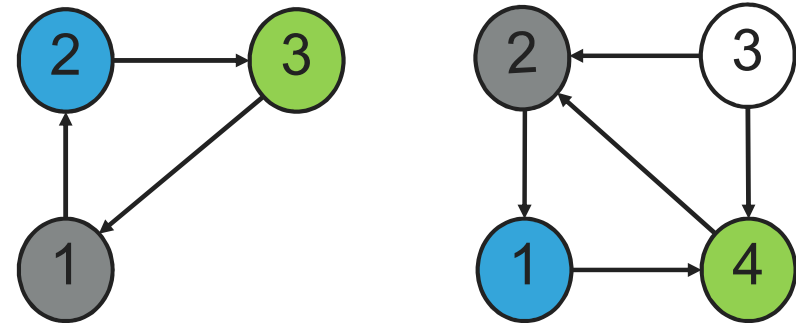
OUTLINE

- ▶ Subgraph Isomorphism
- ▶ VF3: Overview
- ▶ VF3-Light
- ▶ Recent results

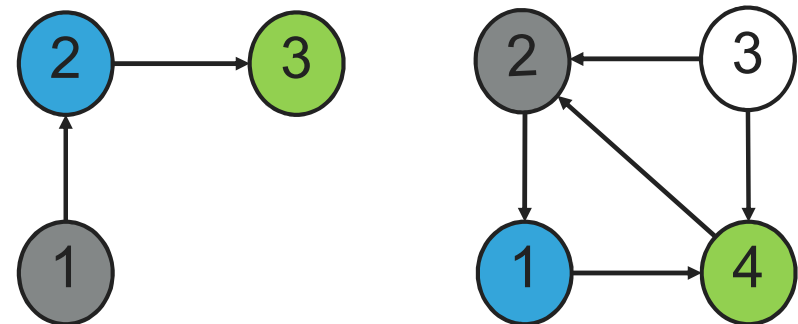
SUBGRAPH ISOMORPHISM

- ▶ Given a graph:
 - ▶ Is it inside another graph?
 - ▶ How many times?
 - ▶ Where?
- ▶ Exact Graph Matching:
 1. Adjacency preserving
 2. Non-adjacency preserving
- ▶ Common applications:
 - ▶ Pattern search or Graph querying
 - ▶ Graph learning algorithms
 - ▶ Graph clustering
- ▶ NP-Complete Problem

Subgraph Isomorphism



Subgraph Monomorphism



VF3

- ▶ The problem is formulated in terms of **State Space Representation**, where each state represents a partial mapping solution;
- ▶ A **depth-first search with backtracking** is used;
- ▶ **Five feasibility rules** are defined in order to prune the state space. These rules take into account:
 - ▶ The consistency of the **current solution**.
 - ▶ Necessary and sufficient conditions.
 - ▶ The consistency of the **“future” solutions** (1- and 2-look-ahead).

VF3 FEASIBILITY SETS AND RULES

- The feasibility rules are evaluated on the feasibility sets

$$\text{ISFEASIBLE}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) =$$

$$F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$F_{la1}(s_c, u_n, v_n) \iff F_{la1}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la1}^q(s_c, u_n, v_n)$$

$$F_{la2}(s_c, u_n, v_n) \iff$$

$$F_{la2}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la2}^q(s_c, u_n, v_n)$$

$$F_{la1}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{P}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$F_{la2}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{V}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{V}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{V}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{V}_2^{c_i}(s_c)|$$

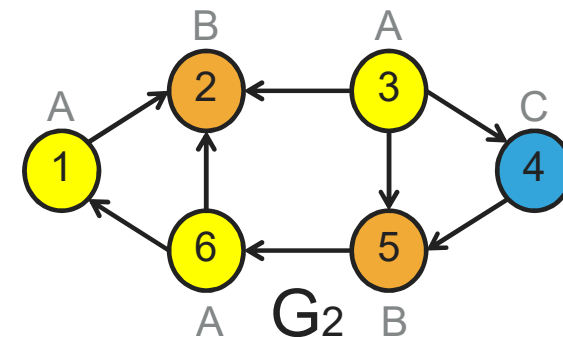
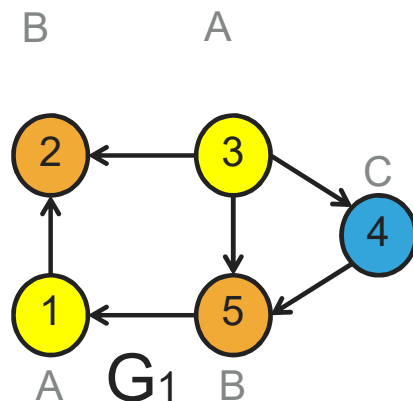
- ▶ **Pattern reordering** and **pre-processing procedures** allow the algorithm:
 - ▶ To be insensitive w.r.t. the permutations.
 - ▶ To compute the information to process the pattern before starting the matching.
- ▶ A **node classification criterion** used:
 - ▶ Nodes are divided in disjointed subsets (feasibility sets)
 - ▶ Nodes in different feasibility sets, they will never be matched
 - ▶ To improve the effectiveness of the feasibility rules.
 - ▶ To speed the selection of candidates to generate new states.

CLASSIFICATION EXAMPLE

- ▶ A simple classification function assigns a different class for each different label in G_1 and G_2

Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2
6	A	c_1



SORTING EXAMPLE

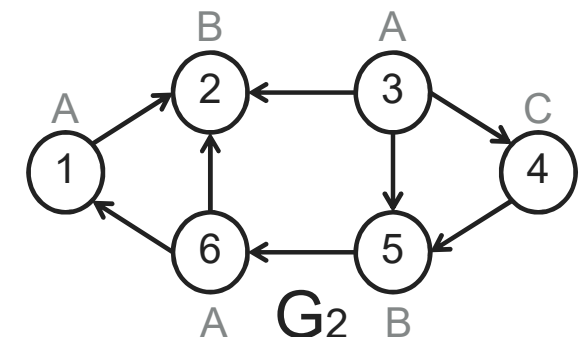
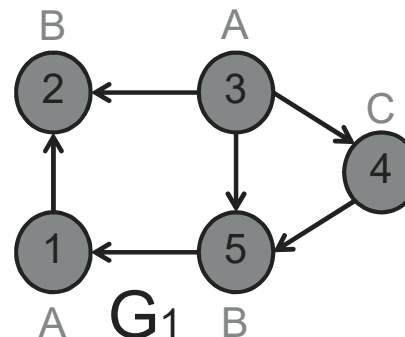
$$P_f(u) = \Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

$$P_f(u) = P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

$Ng = \{3, 5, 4, 2, 1\}$

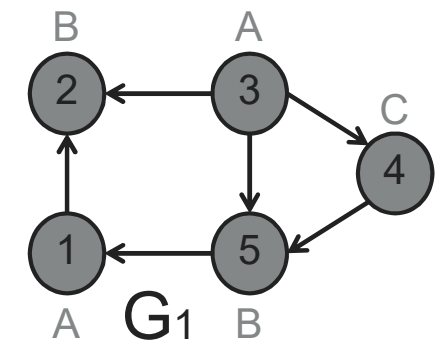
Node	Connections with Ng	degree
1	-	2
2	-	2
3	-	3
4	-	2
5	-	3



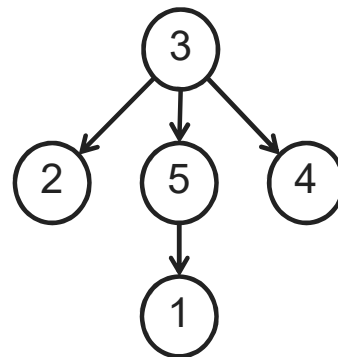
VF3: PREPROCESSING EXAMPLE

Level	Mapped	\tilde{P}_1^{c1}	\tilde{P}_1^{c2}	\tilde{P}_1^{c3}	\tilde{S}_1^{c1}	\tilde{S}_1^{c2}	\tilde{S}_1^{c3}
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}
2	{3,5}	{}	{}	{4}	{1}	{2}	{4}
3	{3,4,5}	{}	{}	{}	{1}	{2}	{}
4	{2,3,4,5}	{1}	{}	{}	{1}	{}	{}
5	{1,2,3,4,5}	{}	{}	{}	{}	{}	{}

$$N_g = \{3, 5, 4, 2, 1\}$$



Parent tree



Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

FROM VF3 TO VF3-LIGHT

- ▶ VF3 has been proved to be the most efficient algorithm to process very large and dense graphs (Carletti et al. 2018)
 - ▶ Complex biological networks (eg. Proteins, interaction networks)
 - ▶ Social network analysis
 - ▶ Knowledge base as graphs
- ▶ Not all the heuristics introduced with VF3 are necessary when dealing with small or sparse graphs.
 - ▶ It is important to find a trade-off between the number of states explored by the algorithm and the time required to process each state.
- ▶ VF3-Light is a lightened version of VF3 where:
 - ▶ Look-ahead rules are removed

FROM VF3 TO VF3-LIGHT

- Look-ahead rules are removed in order to save time while processing a state:

$$\text{ISFEASIBLE}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) =$$

$$F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$F_{la1}(s_c, u_n, v_n) \iff F_{la1}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la1}^q(s_c, u_n, v_n)$$

$$F_{la2}(s_c, u_n, v_n) \iff$$

$$F_{la2}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la2}^q(s_c, u_n, v_n)$$

$$F_{la1}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{P}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$F_{la2}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{\mathcal{V}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{V}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{V}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{V}}_2^{c_i}(s_c)|$$

FROM VF3 TO VF3-LIGHT

- Look-ahead rules are removed in order to save time while processing a state:

$$\text{ISFEASIBLE}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) =$$

$$F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$F_{la1}(s_c, u_n, v_n) \iff F_{la1}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la1}^q(s_c, u_n, v_n)$$

$$F_{la2}(s_c, u_n, v_n) \iff F_{la2}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la2}^q(s_c, u_n, v_n)$$

$$F_{la1}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{P}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$F_{la2}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{\mathcal{V}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{V}}_2^{c_i}(s_c)|$$

$$\wedge |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{V}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{V}}_2^{c_i}(s_c)|$$

FROM VF3 TO VF3-LIGHT

- Look-ahead rules are removed in order to save time while processing a state:

$$\text{ISFEASIBLE}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) =$$

$$F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$F_{la1}(s_c, u_n, v_n) \iff F_{la1}^1(s_c, u_n, v_n) \wedge \dots \wedge F_{la1}^q(s_c, u_n, v_n)$$

$$F_{la1}^i(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge \quad |\mathcal{P}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

$$\wedge \quad |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{P}}_2^{c_i}(s_c)|$$

$$\wedge \quad |\mathcal{S}_1(u_n) \cap \tilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \tilde{\mathcal{S}}_2^{c_i}(s_c)|$$

FROM VF3 TO VF3-LIGHT

- Look-ahead rules are removed in order to save time while processing a state:

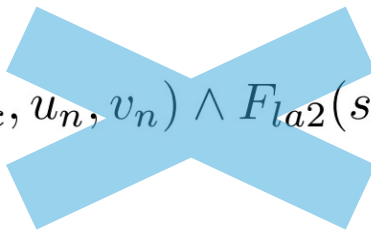
$$\text{ISFEASIBLE}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) =$$

$$F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

FROM VF3 TO VF3-LIGHT

- ▶ VF3-Light is a lightened version of VF3 where:
- ▶ Look-ahead rules are removed in order to save time while processing a state

$$F_t(s_c, u_n, v_n) = F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$


- ▶ Removed the feasibility sets:
 - ▶ Reduced the time required by the pre-processing
 - ▶ Reduced the time required to update the sets at each state
 - ▶ Reduced the memory usage

EXPERIMENTS: DATASETS

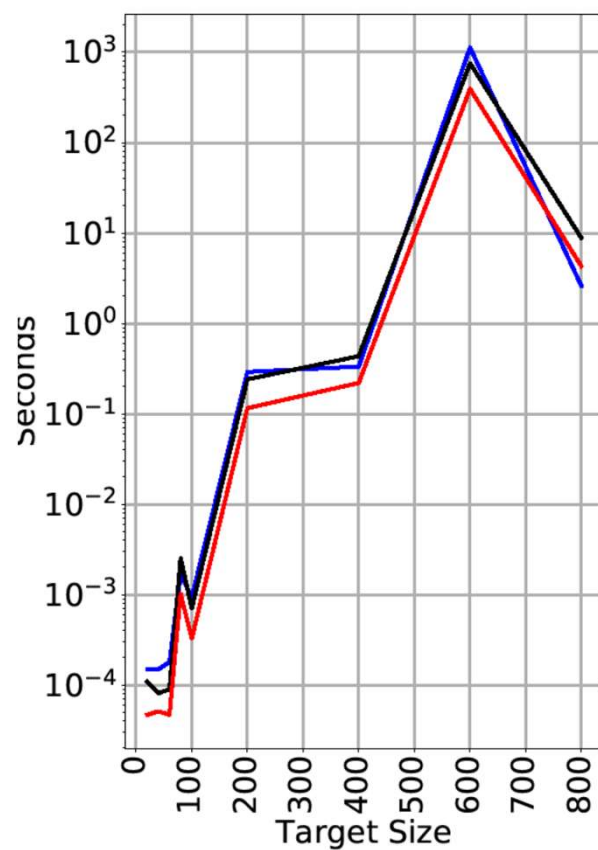
- ▶ Datasets:
 - ▶ Standard MIVIA dataset
 - ▶ MIVIA biological dataset
 - ▶ Solnon Scale Free dataset
- ▶ Algorithms:
 - ▶ VF3
 - ▶ VF3-Light
 - ▶ RI

Dataset	Graphs	Target size	Pattern size	Labels
MIVIA BVG	6000	20 – 1000 nodes	20% of target size	-
MIVIA M2D	4000	16 – 1024 nodes	20% of target size	-
MIVIA M3D	3200	27 – 1000 nodes	20% of target size	-
MIVIA M4D	2000	16 – 1096 nodes	20% of target size	-
MIVIA RAND	3000	20 – 1000 nodes	20% of target size	-
Proteins	300	535 – 10081 nodes	8 – 256	4 – 5
Molecules	10000	8 – 99 nodes	8 – 64	4 – 5
Scale-Free	100	200 – 1000 nodes	90% of target size	-

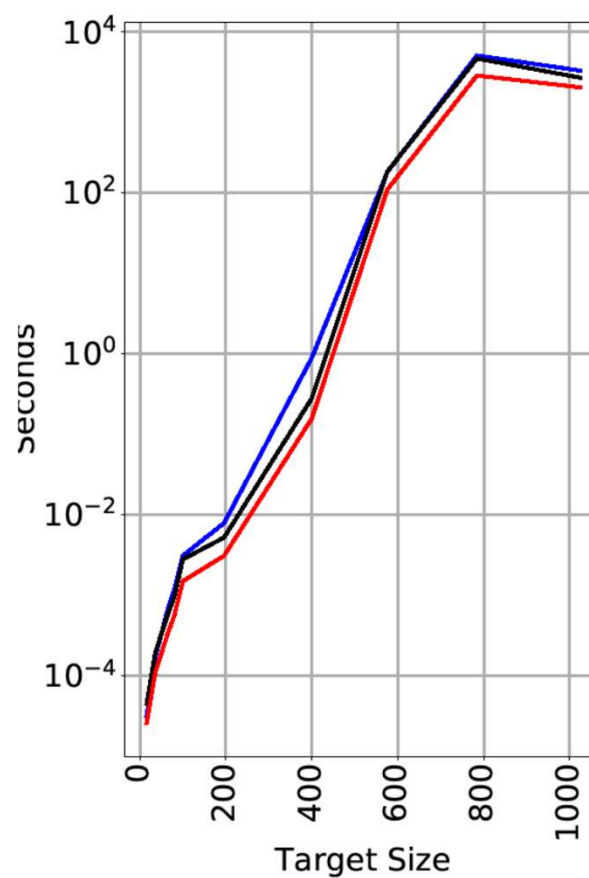
RESULTS

VF3 VF3-Light RI

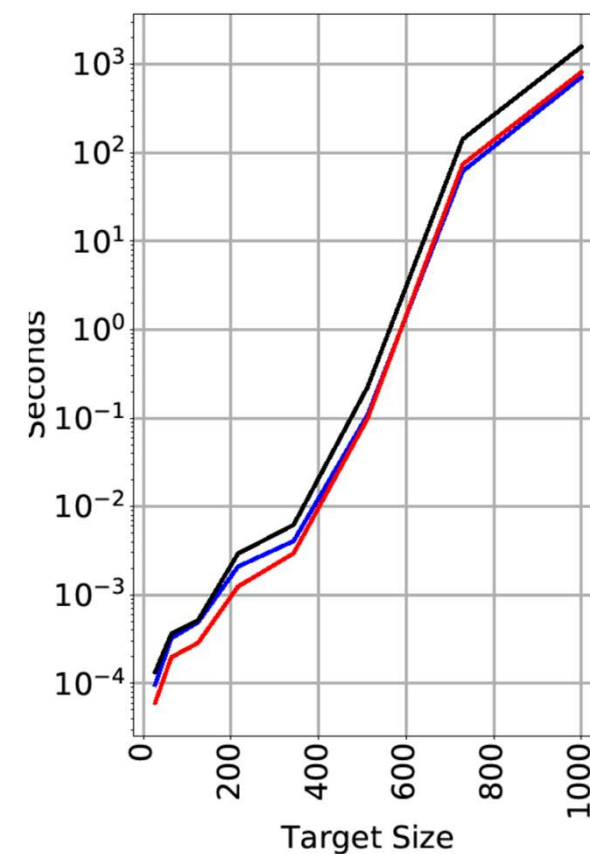
Bounded Valence



Mashes 2D



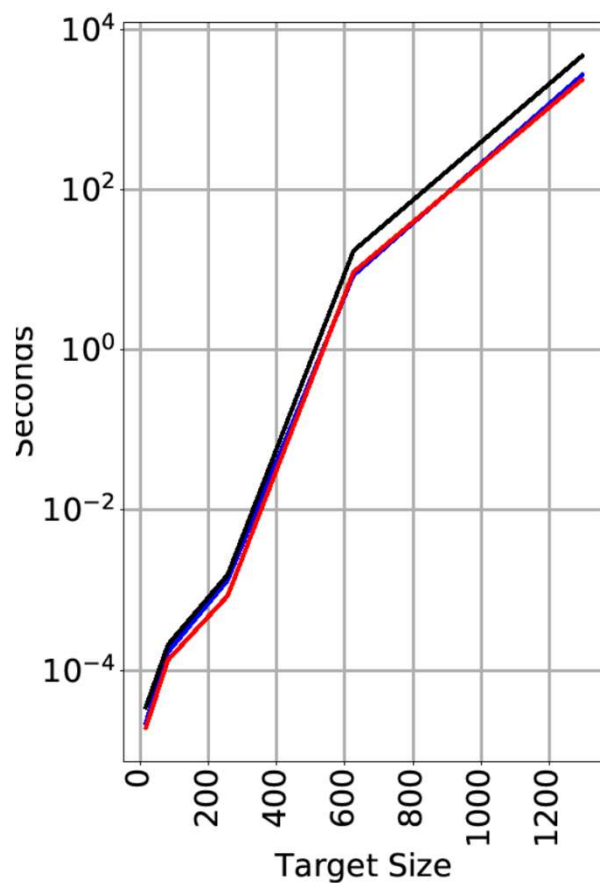
Mashes 3D



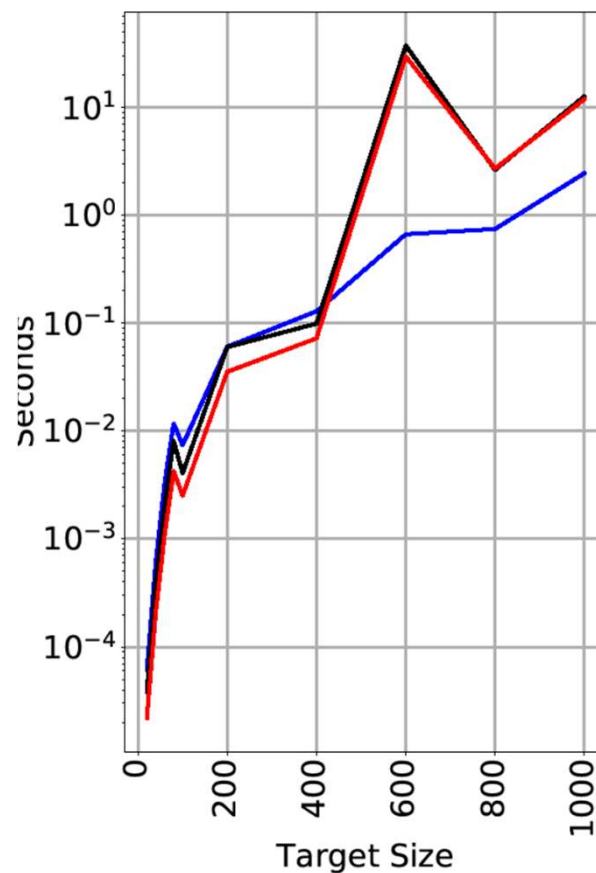
RESULTS

VF3 VF3-Light RI

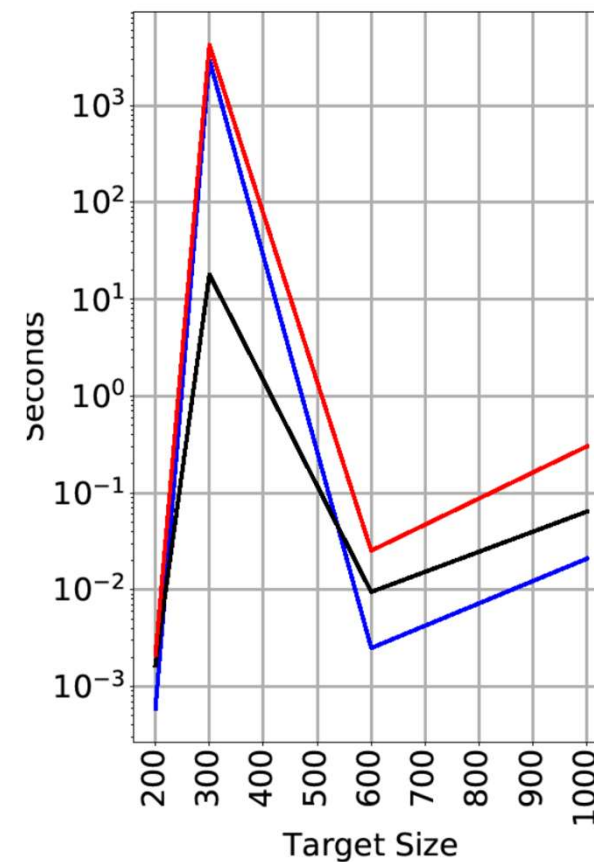
Mashes 4D



Random



Scale Free



RESULTS

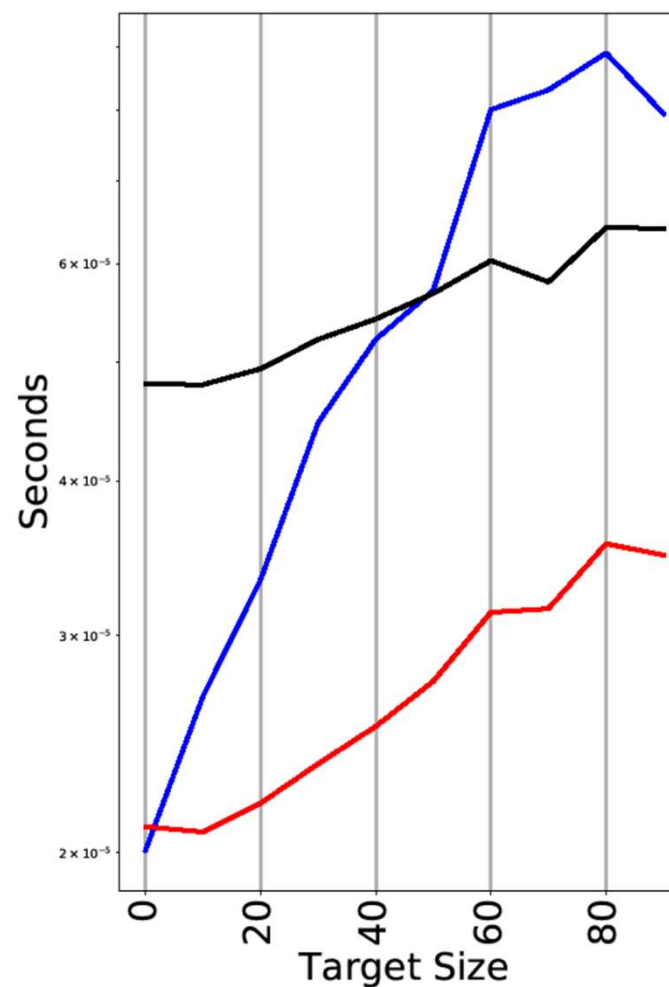
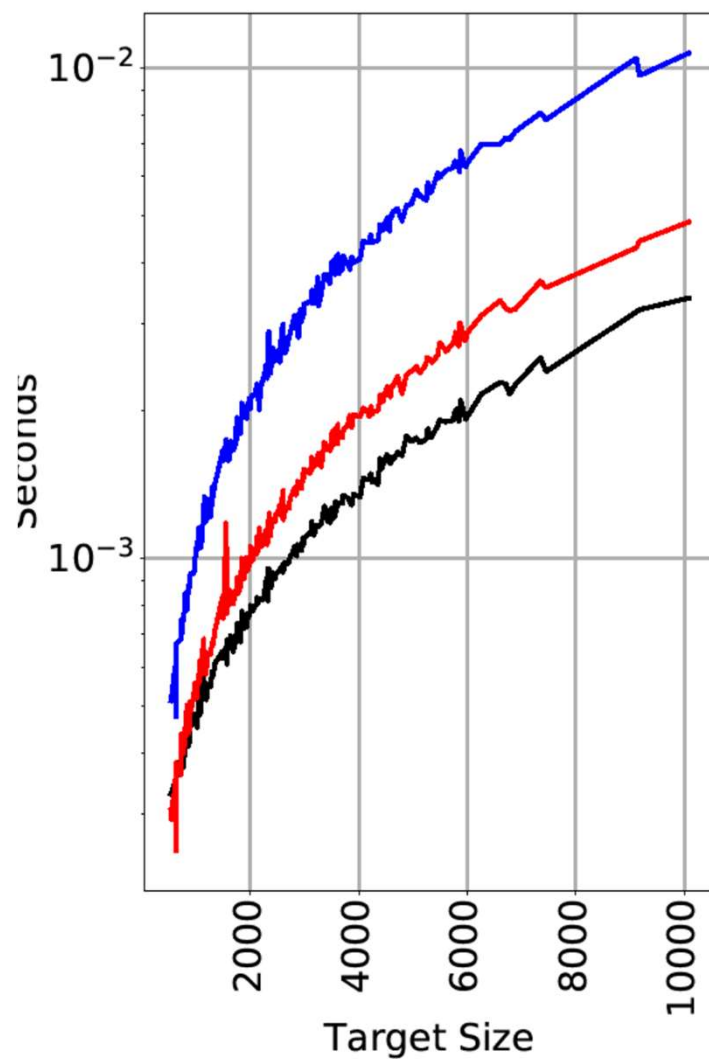
VF3

VF3-Light

RI

Proteins

Molecules



RESULTS

	VF3		VF3-Light		RI	
	Time	Relative Time	Time	Relative Time	Time	Relative Time
BVG	1.41e+05	1.92	7.33e+04	1.00	2.10e+05	2.87
RAND	1.58e+04	12.96	1.33e+04	10.87	1.22e+03	1.00
M2D	9.02e+05	1.63	5.55e+05	1.00	9.76e+05	1.76
M3D	6.89e+05	2.22	3.56e+05	1.15	3.11e+05	1.00
M4D	1.33e+05	1.98	6.73e+04	1.00	7.62e+04	1.13
Molecules	2.25e+01	2.19	1.02e+01	1.0	2.30e+01	2.24
Proteins	1.94e+01	1.0	2.62e+01	1.35	5.69e+01	2.93
Scale-Free	6.32e+02	1.00	1.48e+05	233.65	1.04e+05	164.09

- ▶ **Time**: Time required to find all the solutions.
- ▶ **Relative Time**: normalized algorithm time with respect to the fastest algorithm.

RESULTS

	VF3		VF3-Light		RI	
	Time	Relative Time	Time	Relative Time	Time	Relative Time
BVG	1.41e+05	1.92	7.33e+04	1.00	2.10e+05	2.87
RAND	1.58e+04	12.96	1.33e+04	10.87	1.22e+03	1.00
M2D	9.02e+05	1.63	5.55e+05	1.00	9.76e+05	1.76
M3D	6.89e+05	2.22	3.56e+05	1.15	3.11e+05	1.00
M4D	1.33e+05	1.98	6.73e+04	1.00	7.62e+04	1.13
Molecules	2.25e+01	2.19	1.02e+01	1.0	2.30e+01	2.24
Proteins	1.94e+01	1.0	2.62e+01	1.35	5.69e+01	2.93
Scale-Free	6.32e+02	1.00	1.48e+05	233.65	1.04e+05	164.09

- ▶ VF3 outperforms both VF3-Light and RI on large graphs (that are respectively 35% and almost 200% slower)
- ▶ On small graphs VF3-Light is always faster than VF3.

RESULTS

	Size	VF3		VF3-Light		RI	
		Time	Relative Time	Time	Relative Time	Time	Relative Time
BVG	80	2.54e-03	2.49	1.02e-03	1.00	1.67e-03	1.64
	100	7.06e-04	2.16	3.26e-04	1.00	9.32e-04	2.86
	200	2.41e-01	2.08	1.15e-01	1.00	2.90e-01	2.52
	400	4.34e-01	1.98	2.19e-01	1.00	3.33e-01	1.52
	600	7.54e+02	1.92	3.93e+02	1.00	1.13e+03	2.87
	800	8.82e+00	3.39	4.30e+00	1.65	2.60e+00	1.00

RESULTS

	Size	VF3		VF3-Light		RI	
		Time	Relative Time	Time	Relative Time	Time	Relative Time
RAND	80	8.13e-03	1.91	4.25e-03	1.00	1.18e-02	2.77
	100	4.07e-03	1.61	2.52e-03	1.00	7.40e-03	2.93
	200	6.00e-02	1.69	3.54e-02	1.00	6.04e-02	1.71
	400	9.91e-02	1.37	7.23e-02	1.00	1.29e-01	1.78
	600	3.74e+01	56.12	2.96e+01	44.39	6.66e-01	1.00
	800	2.63e+00	3.53	2.71e+00	3.63	7.45e-01	1.00
	1000	1.26e+01	5.15	1.19e+01	4.85	2.45e+00	1.00

RESULTS

	Size	VF3		VF3-Light		RI	
		Time	Relative Time	Time	Relative Time	Time	Relative Time
M2D	81	9.81e-04	1.72	5.70e-04	1.00	1.22e-03	2.14
	100	2.77e-03	1.87	1.49e-03	1.00	3.08e-03	2.07
	196	5.18e-03	1.69	3.07e-03	1.00	7.84e-03	2.55
	400	2.78e-01	1.78	1.56e-01	1.00	8.84e-01	5.67
	576	1.83e+02	1.67	1.10e+02	1.00	1.81e+02	1.65
	784	4.64e+03	1.63	2.85e+03	1.00	5.05e+03	1.77
	1024	2.68e+03	1.32	2.03e+03	1.00	3.28e+03	1.61
M3D	64	3.64e-04	1.84	1.98e-04	1.00	3.24e-04	1.64
	125	5.19e-04	1.81	2.87e-04	1.00	4.93e-04	1.72
	216	2.93e-03	2.36	1.24e-03	1.00	2.09e-03	1.68
	343	6.21e-03	2.10	2.96e-03	1.00	4.07e-03	1.38
	512	2.25e-01	2.26	9.95e-02	1.00	1.09e-01	1.09
	729	1.43e+02	2.31	7.42e+01	1.20	6.20e+01	1.00
	1000	1.59e+03	2.21	8.20e+02	1.14	7.19e+02	1.00

RESULTS

	Size	VF3		VF3-Light		RI	
		Time	Relative Time	Time	Relative Time	Time	Relative Time
M4D	16	3.46e-05	1.80	1.92e-05	1.00	2.22e-05	1.16
	81	2.09e-04	1.55	1.35e-04	1.00	1.69e-04	1.26
	256	1.56e-03	1.83	8.51e-04	1.00	1.33e-03	1.57
	625	1.72e+01	2.02	9.34e+00	1.09	8.53e+00	1.00
	1296	4.68e+03	1.99	2.36e+03	1.00	2.70e+03	1.15

CONCLUSIONS

- ▶ VF3-Light, a subgraph isomorphism algorithm obtained by removing some of the heuristics used in VF3.
- ▶ The algorithm is faster in visiting each state
 - ▶ A larger number of states may need to be visited to find the solutions.
- ▶ On small graphs VF3-Light is able to outperform VF3

