technische universität
dortmund

Reconsideration and Extension of Cartesian Genetic Programming

Dissertation defense

Roman Kalkreuth

Computational Intelligence Research Group
Chair XI - Algorithm Engineering
Department of Computer Science
TU Dortmund University

27/07/2021

Conference papers (1):

Roman Kalkreuth, Günter Rudolph, and Jörg Krone. Improving Convergence in Cartesian Genetic Programming Using Adaptive Crossover, Mutation and Selection. In: IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7-10, 2015, pages 1415–1422. IEEE, 2015. DOI: 10.1109/SSCI.2015.20

Roman Kalkreuth, Günter Rudolph, and Andre Droschinsky. A New Subgraph Crossover for Cartesian Genetic Programming. In: Genetic Programming - 20th European Conference, EuroGP 2017, Amsterdam, The Netherlands, April 19-21, 2017, Proceedings, Lecture Notes in Computer Science, Volume 10196, pages 294–310, 2017. DOI: 10.1007/978-3-319-55696-3_19

Paul Kaufmann and Roman Kalkreuth. An Empirical Study on the Parametrization of Cartesian Genetic Programming. In: Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15-19, 2017, Companion Material Proceedings, pages 231–232. ACM, 2017. DOI: 10.1145/3067695.3075980

Paul Kaufmann and Roman Kalkreuth. Parametrizing Cartesian Genetic Programming: An Empirical Study. In: KI 2017: Advances in Artificial Intelligence - 40th Annual German Conference on AI, Dortmund, Germany, September 25-29, 2017, Proceedings, Lecture Notes in Computer Science, Volume 10505, pages 316–322. Springer, 2017. DOI: 10.1007/978-3-319-67190-1_26

Conference papers (2):

Jakub Husa and Roman Kalkreuth. A Comparative Study on Crossover in Cartesian Genetic Programming. In: Genetic Programming - 21st European Conference, EuroGP 2018, Parma, Italy, April 4-6, 2018, Proceedings, Lecture Notes in Computer Science, volume 10781, pages 203–219. Springer, 2018. DOI: 10.1007/978-3-319-77553-1_13

Roman Kalkreuth and Andre Droschinsky. On the Time Complexity of Simple Cartesian Genetic Programming. In: Proceedings of the 11th International Joint Conferenceon Computational Intelligence, IJCCI 2019, Vienna, Austria, September 17-19, 2019, pages 172–179. ScitePress, 2019. DOI: 10.5220/0008070201720179 **(Best Poster Nomination)**

Roman Kalkreuth. Two new Mutation Techniques for Cartesian Genetic Programming. In: Proceedings of the 11th International Joint Conference on Computational Intelligence, IJCCI 2019, Vienna, Austria, September 17-19, 2019, pages 82–92. ScitePress, 2019. DOI: 10.5220/0008070100820092 **(Best Student Paper Nomination)**

Roman Kalkreuth. A Comprehensive Study on Subgraph Crossover in Cartesian Genetic Programming. In: Proceedings of the 12th International Joint Conferenceon Computational Intelligence, IJCCI 2020, Budapest, Hungary, November 2-4, 2020, pages 59–70. SCITEPRESS, 2020. DOI:
10.5220/0010110700590070

Book chapters:

Roman Kalkreuth: An Empirical Study on Insertion and Deletion Mutation in Cartesian Genetic Programming, In: Computational Intelligence: 11th International Joint Conference, IJCCI 2019, Vienna, Austria, September 17–19, 2019, Revised Selected Papers, Springer International Publishing, 2021. DOI:
10.1007/978-3-030-70594-7_4

Outline

Genetic Programming (GP)

- Genetic Programming is a search heuristic

Genetic Programming (GP)

- Genetic Programming is a search heuristic
- Evolutionary algorithm-based method for automatic derivation of computer programs

Genetic Programming (GP)

- Genetic Programming is a search heuristic
- Evolutionary algorithm-based method for automatic derivation of computer programs
- Inspired by Charles Darwin's theory of evolution[1]

[1] Charles Darwin: *On the Origin of Species by Means of Natural Selection,1859*

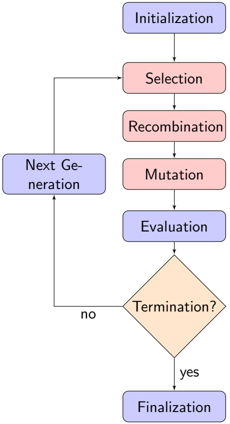
Genetic Programming (GP)

Definition (Genetic Programming)

Let Θ be a population of $|\Theta|$ individuals and let Ω be the population of the following generation:

- Each individual is represented with a **genetic program** and a **fitness value**.
- Genetic Programming transforms $\Theta \mapsto \Omega$ by the adaptation of **selection**, **recombination** and **mutation**.

Genetic Programming (GP)



Termination criteria $\rightarrow$ predefined fitness reached or budget of generations exceeded

Genetic Programming (GP)

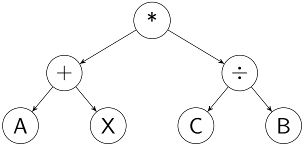
Definition (Genetic Program)

A genetic programm $\mathcal{P}$ is an element of $\mathcal{T} \times \mathcal{F} \times \mathcal{E}$:

- $\mathcal{F}$ is a finite non-empty set of functions
- $\mathcal{T}$ is a finite non-empty set of terminals
- $\mathcal{E}$ is a finite non-empty set of edges

Let $\phi : \mathcal{P} \mapsto \Psi$ be a decode function which maps $\mathcal{P}$ to a phenotype Ψ

Genetic Programming (GP)



$$\mathcal{F} = \{ + , - , *, \div \}$$

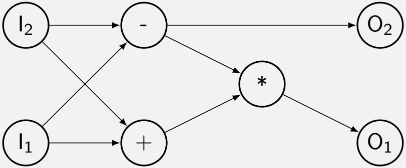
$$\mathcal{T} = \{ A, X, C, B \}$$

$$\mathcal{E} = \text{Edges}$$

$$\Psi = (A + X) * (C \div B)$$

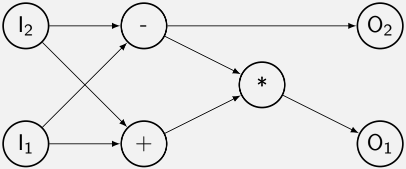
Cartesian Genetic Programming (CGP)

- Genetic Programming → traditionally tree representation



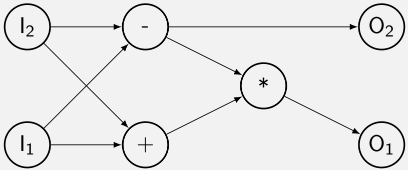
Cartesian Genetic Programming (CGP)

- Genetic Programming → traditionally tree representation
- Cartesian Genetic Programming → graph representation



Cartesian Genetic Programming (CGP)

- Genetic Programming → traditionally tree representation
- Cartesian Genetic Programming → graph representation
- Extension of Genetic Programming



Cartesian Genetic Programming (CGP)

- Program representation $\rightarrow$ acyclic and directed graph

Cartesian Genetic Programming (CGP)

- Program representation $\rightarrow$ acyclic and directed graph
- Genotype-phenotype mapping $\rightarrow$ encoding-decoding of the graph

Cartesian Genetic Programming (CGP)

- Program representation $\rightarrow$ acyclic and directed graph
- Genotype-phenotype mapping $\rightarrow$ encoding-decoding of the graph
- Predominantly used without recombination $\rightarrow$ $(1+\lambda)$ selection scheme

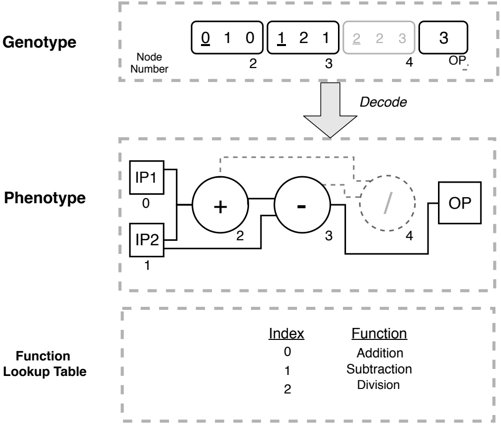
Cartesian Genetic Programming (CGP)

Definition (Cartesian Genetic Program)

A cartesian genetic program $\mathcal{P}$ is an element of $\mathcal{N}_\mathsf{i} \times \mathcal{N}_\mathsf{f} \times \mathcal{N}_\mathsf{o} \times \mathcal{F}$:

- $\mathcal{N}_\mathsf{i}$ is a finite non-empty set of input nodes
- $\mathcal{N}_\mathsf{f}$ is a finite set of function nodes
- $\mathcal{N}_\mathsf{o}$ is a finite non-empty set of output nodes
- $\mathcal{F}$ is a finite non-empty set of functions

Cartesian Genetic Programming (CGP)



State of Scientific Knowledge in CGP

- One-sided and incomplete

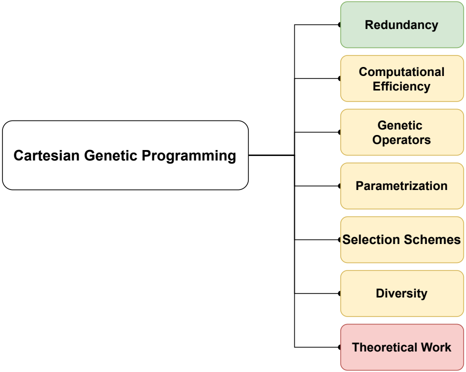
State of Scientific Knowledge in CGP

- One-sided and incomplete
- Empirical findings $\rightarrow$ illegitimate generalization

State of Scientific Knowledge in CGP

- One-sided and incomplete
- Empirical findings $\rightarrow$ illegitimate generalization
- Overemphasis on Boolean function learning

State of Scientific Knowledge in CGP



State of Scientific Knowledge in CGP

Urgent open issues:

- Theoretical work $\rightarrow$ lack of formalism and runtime analysis

[2] Julian Miller: *Cartesian Genetic Programming: its status and future*, Genetic Programming and Evolvable Machines, 2020

State of Scientific Knowledge in CGP

Urgent open issues:

- Theoretical work $\rightarrow$ lack of formalism and runtime analysis
- Genetic variation $\rightarrow$ recombination and mutation[2]

[2] Julian Miller: *Cartesian Genetic Programming: its status and future*, Genetic Programming and Evolvable Machines, 2020

State of Scientific Knowledge in CGP

Urgent open issues:

- Theoretical work $\rightarrow$ lack of formalism and runtime analysis
- Genetic variation $\rightarrow$ recombination and mutation[2]
- Recombination $\rightarrow$ long standing open question in CGP

[2] Julian Miller: *Cartesian Genetic Programming: its status and future*, Genetic Programming and Evolvable Machines, 2020

State of Scientific Knowledge in CGP

Urgent open issues:

- Theoretical work → lack of formalism and runtime analysis
- Genetic variation → recombination and mutation[2]
- Recombination → long standing open question in CGP
- Parametrization → illegitimate generalization

[2] Julian Miller: *Cartesian Genetic Programming: its status and future*, Genetic Programming and Evolvable Machines, 2020

Recombination in CGP: A two decades-old open issue

1999	Miller: Genotypic crossover
2007	Clegg, Walker, Miller: Arithmetic crossover
2008	Kaufmann, Platzner: Cone-based crossover (Modular CGP)
2015	Kalkreuth, Rudolph, Krone: Adaptive arithmetic crossover
2017	Kalkreuth, Rudolph, Droschinsky: **Subgraph crossover**
2018	Husa, Kalkreuth: **Block crossover**
2020	Kalkreuth: Comparative study on crossover

Thesis Contributions

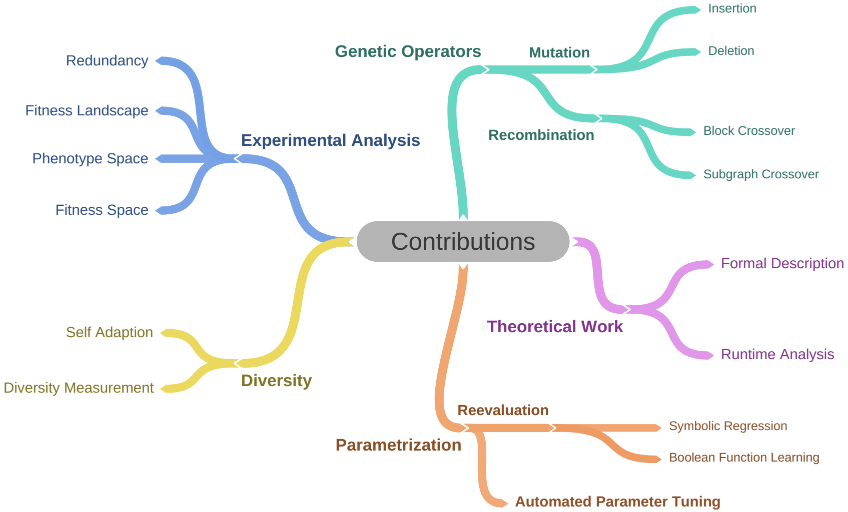
- Genetic variation $\rightarrow$ new operators for recombination and mutation

Thesis Contributions

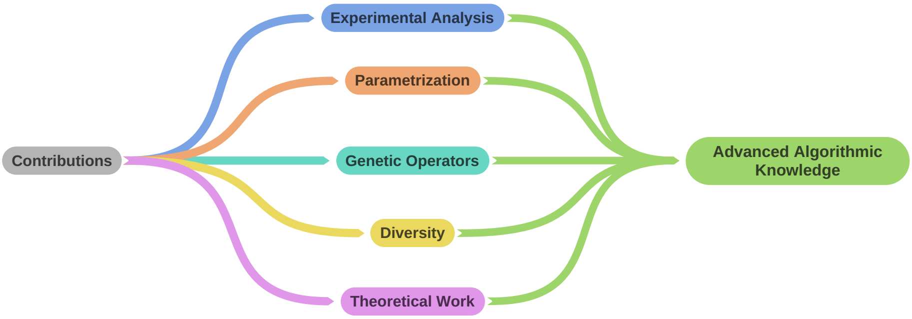
- Genetic variation $\rightarrow$ new operators for recombination and mutation
- Parametrization studies $\rightarrow$ covering different problem domains

Thesis Contributions

- Genetic variation $\rightarrow$ new operators for recombination and mutation
- Parametrization studies $\rightarrow$ covering different problem domains
- First runtime analysis and formal description of CGP


Contributions
Genetic Operators
Mutation
Insertion
Deletion
Recombination
Block Crossover
Subgraph Crossover
Experimental Analysis
Redundancy
Fitness Landscape
Phenotype Space
Fitness Space
Diversity
Self Adaption
Diversity Measurement
Theoretical Work
Formal Description
Runtime Analysis
Parametrization
Reevaluation
Symbolic Regression
Boolean Function Learning
Automated Parameter Tuning

Thesis Contributions



Advanced Genetic Operators in CGP

- Genotypic variation $\rightarrow$ traditional approach

Advanced Genetic Operators in CGP

- Genotypic variation $\rightarrow$ traditional approach
- Phenotypic variation $\rightarrow$ new approach

Advanced Genetic Operators in CGP

- Genotypic variation → traditional approach
- Phenotypic variation → new approach
- Inspired by Epigenetics and Lamarckian thought[3]

[3] Jean-Baptiste Lamarck : *Zoological Philosophy: An Exposition with Regard to the Natural History of Animals*, 1809

The Subgraph Crossover

- Underlying idea $\rightarrow$ subtree crossover found in tree-based GP

The Subgraph Crossover

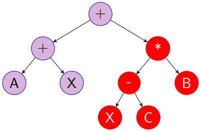
- Underlying idea $\rightarrow$ subtree crossover found in tree-based GP
- Joins and links subgraphs of two selected parents

The Subgraph Crossover

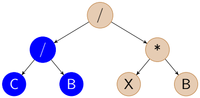
- Underlying idea $\rightarrow$ subtree crossover found in tree-based GP
- Joins and links subgraphs of two selected parents
- Produces one offspring

Inspiration: Subtree Crossover (GP)

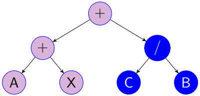
(a) First Parent



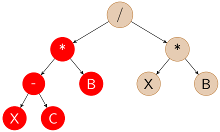
(b) Second Parent



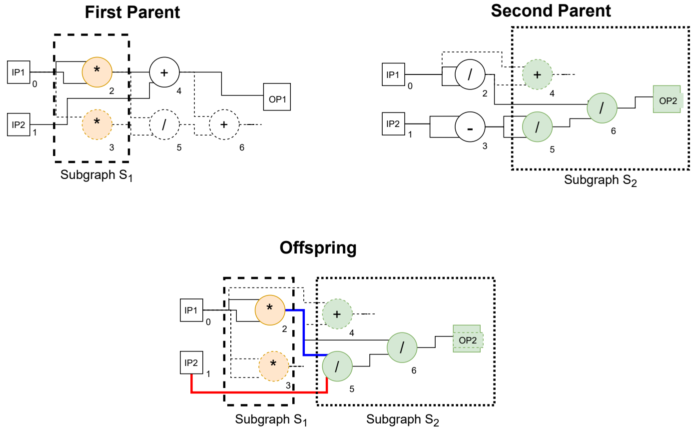
(c) First Offspring



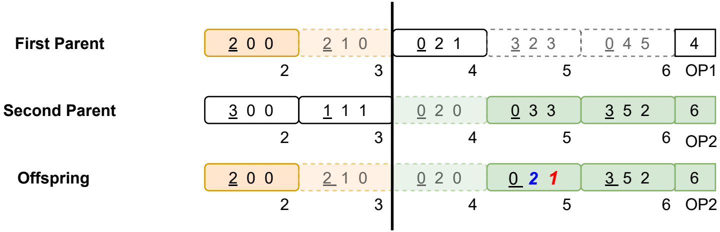
(d) Second Offspring



The Subgraph Crossover (CGP)



The Subgraph Crossover (CGP)



The Block Crossover

- Selects and swaps blocks of function genes

The Block Crossover

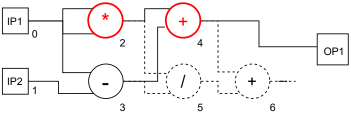
- Selects and swaps blocks of function genes
- Phenotypic variation $\rightarrow$ only active function genes

The Block Crossover

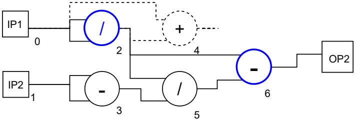
- Selects and swaps blocks of function genes
- Phenotypic variation $\rightarrow$ only active function genes
- Produces two offsprings

The Block Crossover

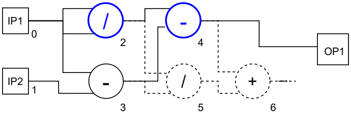
First Parent



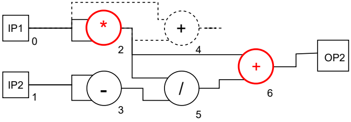
Second Parent



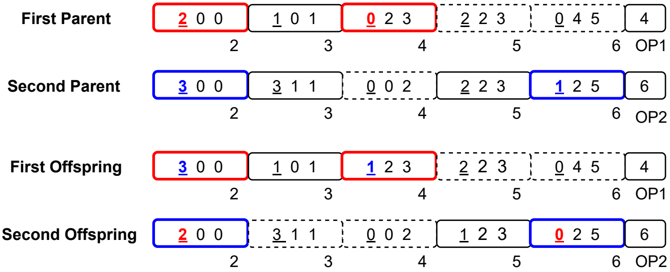
First Offspring



Second Offspring



The Block Crossover



Evaluation

Experimental Setup

- Benchmark problems $\rightarrow$ diverse set of Boolean function and **symbolic regression problems**[4]

[4] McDermott et al.: *Genetic Programming Needs Better Benchmarks*, GECCO '12: Proceedings of the Genetic and Evolutionary Computation Conference, 2012

Evaluation

Experimental Setup

- Benchmark problems → diverse set of Boolean function and **symbolic regression problems**[4]
- Search performance evaluation → number of fitness evaluations or best fitness value

[4] McDermott et al.: *Genetic Programming Needs Better Benchmarks*, GECCO '12: Proceedings of the Genetic and Evolutionary Computation Conference, 2012

Evaluation

Experimental Setup

- Benchmark problems → diverse set of Boolean function and **symbolic regression problems**[4]
- Search performance evaluation → number of fitness evaluations or best fitness value
- Statistical validity → 100 runs per problem

[4] McDermott et al.: *Genetic Programming Needs Better Benchmarks*, GECCO '12: Proceedings of the Genetic and Evolutionary Computation Conference, 2012

Evaluation

List of evaluated CGP algorithms

Algorithm	Description
$(1+\lambda)$-CGP	Standard $(1+\lambda)$-CGP algorithm
$(1+\lambda)$-CGP-ID $\star$	$(1+\lambda)$-CGP with insertion and deletion mutation
$(\mu+\lambda)$-CGP (Subgraph) $\star$	$(\mu+\lambda)$-CGP with subgraph crossover
$(\mu+\lambda)$-CGP (Block) $\star$	$(\mu+\lambda)$-CGP with block crossover
$(\mu+\lambda)$-CGP-ID (Subgraph) $\star$	$(\mu+\lambda)$-CGP with insertion/deletion mutation and subgraph crossover
$(\mu+\lambda)$-CGP-ID (Block) $\star$	$(\mu+\lambda)$-CGP with insertion/deletion mutation and block crossover
Canonical-CGP (Subgraph) $\star$	Canonical EA with subgraph crossover and tournament selection
Canonical-CGP (Block) $\star$	Canonical EA with block crossover and tournament selection
Real-valued CGP	Real-valued CGP with decimal representation
Adaptive real-valued CGP $\star$	Real-valued CGP algorithm with self-adaptive strategy

$\star \longrightarrow$ new

Evaluation

Symbolic Regression Benchmarks

Problem	Objective Function	Vars	Training Set	Function Set
Koza-1	$x^4 + x^3 + x^2 + x$	1	U[-1,1,20]	Koza*
Koza-2	$x^5 - 2x^3 + x$	1	U[-1,1,20]	Koza
Koza-3	$x^6 - 2x^4 + x^2$	1	U[-1,1,20]	Koza
Nguyen-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	1	U[-1,1,20]	Koza
Nguyen-5	$\sin(x^2)\cos(x) - 1$	1	U[-1,1,20]	Koza
Nguyen-6	$\sin(x) + \sin(x + x^2)$	1	U[-1,1,20]	Koza
Nguyen-7	$\ln(x+1) + \ln(x^2+1)$	1	U[0,2,20]	Koza
Keijzer-6	$\sum_i^x 1/i$	1	E[1,50,1]	Keijzer†
Pagie-1	$1/(1+x^{-4}) + 1/(1+y^{-4})$	2	E[-5,5,0.4]	Koza

*Koza $= \{ +, -, *, /, \sin, \cos, \ln(|n|), e^n \}$

†Keijzer $= \{ +, *, n^{-1}, -n, \sqrt{n} \}$

Evaluation

Problem	Algorithm	Mean Fitness Evaluations	Standard deviation	Median
Koza-1	$(1+\lambda)$-CGP	5734303	12623995	1210536
	Canonical-CGP (Subgraph)	997014	1164434	644976
	Canonical-CGP (Block)	351981	775846	61776
Koza-2	$(1+\lambda)$-CGP	4846164	1042512	1042512
	Canonical-CGP (Subgraph)	768608	341328	341328
	Canonical-CGP (Block)	248863	60048	60048
Koza-3	$(1+\lambda)$-CGP	742311	1846722	118576
	Canonical-CGP (Subgraph)	87657	129750	46680
	Canonical-CGP (Block)	95869	326758	14184
Nguyen-4	$(1+\lambda)$-CGP	2327772	11592888	29648
	Canonical-CGP (Subgraph)	28464	30350	16320
	Canonical-CGP (Block)	76736	167352	31176
Nguyen-5	$(1+\lambda)$-CGP	3831520	9956078	721312
	Canonical-CGP (Subgraph)	161028	271872	80160
	Canonical-CGP (Block)	314377	544622	123120
Nguyen-6	$(1+\lambda)$-CGP	10461344	22773705	445672
	Canonical-CGP (Subgraph)	356406	1087258	24432
	Canonical-CGP (Block)	227955	470109	59040

Evaluation

Problem	Algorithm	Total Runtime hh:mm:ss	Unfinished Runs
Koza-1	$(1+\lambda)$-CGP	02:25:48	2
	Canonical-CGP (Subgraph)	00:12:47	0
	Canonical-CGP (Block)	00:09:08	0
Koza-2	$(1+\lambda)$-CGP	02:02:04	0
	Canonical-CGP (Subgraph)	00:15:53	0
	Canonical-CGP (Block)	00:07:54	0
Koza-3	$(1+\lambda)$-CGP	00:17:45	0
	Canonical-CGP (Subgraph)	00:02:01	0
	Canonical-CGP (Block)	00:02:59	0
Nguyen-4	$(1+\lambda)$-CGP	04:00:16	2
	Canonical-CGP (Subgraph)	00:04:42	0
	Canonical-CGP (Block)	00:04:47	0
Nguyen-5	$(1+\lambda)$-CGP	01:42:24	1
	Canonical-CGP (Subgraph)	00:05:57	0
	Canonical-CGP (Block)	00:17:46	0
Nguyen-6	$(1+\lambda)$-CGP	04:33:37	7
	Canonical-CGP (Subgraph)	00:06:56	0
	Canonical-CGP (Block)	00:07:38	0

Evaluation

Problem	Algorithm	Mean Best Fitness value	Standard deviation	Median
Nguyen-7	$(1+\lambda)$-CGP	0.71	0.45	0.67
	Canonical-CGP (Subgraph)	0.60	0.35	0.60
	Canonical-CGP (Block)	0.72	0.52	0.65
Keijzer-6	$(1+\lambda)$-CGP	3.38	2.52	3.03
	Canonical-CGP (Subgraph)	2.81	1.13	2.90
	Canonical-CGP (Block)	3.71	2.28	3.15
Pagie-1	$(1+\lambda)$-CGP	120.75	44.95	120.91
	Canonical-CGP (Subgraph)	98.52	50.95	85.31
	Canonical-CGP (Block)	119.97	47.15	115.41

Conclusion

The work of my thesis...

- introduced advanced phenotypic variation in standard CGP

Conclusion

The work of my thesis...

- introduced advanced phenotypic variation in standard CGP
- demonstrated the feasibilities of recombination in CGP

Conclusion

The work of my thesis...

- introduced advanced phenotypic variation in standard CGP
- demonstrated the feasibilities of recombination in CGP
- gave significant insight into the search mechanisms of CGP
- had already a directional impact in graph-based GP[5]

Conclusion

The work of my thesis...

- introduced advanced phenotypic variation in standard CGP
- demonstrated the feasibilities of recombination in CGP
- gave significant insight into the search mechanisms of CGP
- had already a directional impact in graph-based GP[5]

[5] Aktinson et al. : *Evolving graphs with horizontal gene transfer*, GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference, 2019

Conclusion

A first runtime analysis of $(1+1)$-CGP...

- was performed on two artificial problems

Conclusion

A first runtime analysis of $(1+1)$-CGP...

- was performed on two artificial problems
- utilized Drift Analysis[6]

[6] Doerr et al. : *Multiplicative Drift Analysis*, Algorithmica 64, 2011

Conclusion

A first runtime analysis of $(1+1)$-CGP...

- was performed on two artificial problems
- utilized Drift Analysis[6]
- proved that SUM is solved in expected time $\Theta(n \log n)$

[6] Doerr et al. : *Multiplicative Drift Analysis*, Algorithmica 64, 2011

Conclusion

A first runtime analysis of $(1+1)$-CGP...

- was performed on two artificial problems
- utilized Drift Analysis[6]
- proved that SUM is solved in expected time $\Theta(n \log n)$
- proved an upper bound of $\mathcal{O}(n^2 \log n)$ and a lower bound of $\Omega(n^2)$ for COUNTING OPERATORS

[6] Doerr et al. : *Multiplicative Drift Analysis*, Algorithmica 64, 2011

technische universität dortmund

Conclusion

The work of my thesis...

- has already been internationally recognized and acknowledged

New mutation operators and search algorithms

- Kalkreuth recently (arXiv 2018) investigated some new mutation operators: insertion and deletion
  - Insertion: chooses an inactive node and changes one or more connection genes in the genotype to make it active.
  - Deletion: alters connections to an active node so that the node becomes inactive.
  - He examined the impact of the new mutation operators (operating together with the standard point mutation) on three Boolean benchmarks and a suite of symbolic regression problems. On all problems, the use of the two operators gave improved performance.
- Kaufmann and Kalkreuth (2020) extensively investigated various parameters in CGP for effectiveness. Using Goldman's single active mutation strategy, they found
  - 1+1-ES is usually the most efficient algorithm
  - Randomizing all inactive genes before an active gene mutation works well.
  - Mutation of function genes is unnecessary

Crossover or not?

- Recombination doesn't seem to add anything *(Miller 1999, "An empirical study…")*
- However if there are multiple chromosomes with independent fitness assessment then it helps a LOT (*Walker, Miller, Cavill 2006, Walker, Völk, Smith, Miller, 2009)*
- Some work using a floating point representation of CGP has suggested that crossover might be useful (*Clegg, Walker, Miller 2007)*
- Cone-based crossover with modular CGP (Kaufmann, Platzner 2008)
- Kalkreuth has recently investigated sub-graph crossover where *active* nodes are swapped either side of single-point crossover (Kalkreuth 2017)

Source: Cartesian Genetic Programming Tutorial - IEEE Congress of Evolutionary Computation (2021) by Dr. Julian Miller (Founder of CGP)

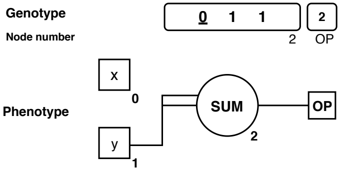
"It is known that every science must have its philosophy, and that it cannot make real progress in any other way."

"If the philosophy of science is neglected her progress will be unreal, and the entire work will remain imperfect."

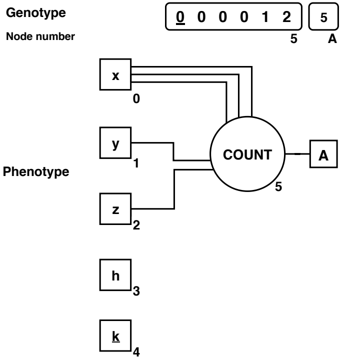
Jean-Baptiste Lamarck: *Zoological Philosophy*
Chapter II: *Importance of the Consideration of Affinities*

,

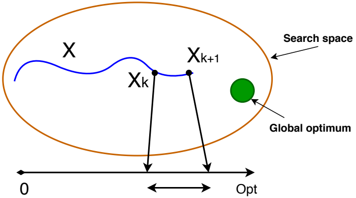
The SUM problem



The COUNTING OPERATOR problem



Drift analysis



$$\Delta_k := g(X_{k+1}) - g(X_k)$$

Active Function Node Range

Problem	Algorithm	Mean Active Function Node Range	Median
Koza-1	$(1+\lambda)$-CGP	2.98	3
	Canonical-CGP (Subgraph)	6.72	7
Koza-2	$(1+\lambda)$-CGP	3.98	4
	Canonical-CGP (Subgraph)	6.72	7
Koza-3	$(1+\lambda)$-CGP	3.59	4
	Canonical-CGP (Subgraph)	6.71	7
Nguyen-4	$(1+\lambda)$-CGP	3,19	3
	Canonical-CGP (Subgraph)	6.05	6
Nguyen-5	$(1+\lambda)$-CGP	3.47	3
	Canonical-CGP (Subgraph)	6.14	6
Nguyen-6	$(1+\lambda)$-CGP	2.91	3
	Canonical-CGP (Subgraph)	5.54	6

Evaluation (Boolean function problems)

Problem	Algorithm	Mean Fitness Evaluations	Standard deviation	Median
Parity-Even-3	$(1+\lambda)$-CGP	2305	3409	1437
	$(1+\lambda)$-CGP-ID	**1208**	**841**	**1047**
	$(\mu+\lambda)$-CGP-ID (Subgraph)	1294	927	1025
Parity-Even-4	$(1+\lambda)$-CGP	9615	6228	7703
	$(1+\lambda)$-CGP-ID	**5942**	**4008**	**4983**
	$(\mu+\lambda)$-CGP-ID (Subgraph)	6083	3376	5176
Parity-Even-5	$(1+\lambda)$-CGP	32099	28146	26037
	$(1+\lambda)$-CGP-ID	**21781**	**12696**	**18142**
	$(\mu+\lambda)$-CGP-ID (Subgraph)	25142	14411	2208
Parity-Even-6	$(1+\lambda)$-CGP	92506	62113	77795
	$(1+\lambda)$-CGP-ID	**69167**	**42159**	**55786**
	$(\mu+\lambda)$-CGP-ID (Subgraph)	74848	53501	58689
Parity-Even-7	$(1+\lambda)$-CGP	263711	171045	239777
	$(1+\lambda)$-CGP-ID	**243992**	**178689**	**172468**
	$(\mu+\lambda)$-CGP (Subgraph)	201532	131936	172038
Parity-Even-8	$(1+\lambda)$-CGP	643263	347129	591060
	$(1+\lambda)$-CGP-ID	433957	285093	332100
	$(\mu+\lambda)$-CGP (Subgraph)	**422444**	**273254**	**315086**

Evaluation (Boolean function problems)

Problem	Algorithm	Mean Fitness Evaluations	Standard deviation	Median
Adder-1Bit	$(1+\lambda)$-CGP	5226	5836	3215
	$(1+\lambda)$-CGP-ID	3879	3166	2827
	$(\mu+\lambda)$-CGP (Subgraph)	**3631**	**3228**	**2605**
Adder-2Bit	$(1+\lambda)$-CGP	86168	84130	58135
	$(1+\lambda)$-CGP-ID	**62574**	**51022**	**42831**
	$(\mu+\lambda)$-CGP (Subgraph)	73312	62167	50795
Adder-3Bit	$(1+\lambda)$-CGP	293639	232145	222136
	$(1+\lambda)$-CGP-ID	270504	210430	215443
	$(\mu+\lambda)$-CGP (Subgraph)	266449	153200	252469
Multiplier-2Bit	$(1+\lambda)$-CGP	7381	7135	4988
	$(1+\lambda)$-CGP-ID	6207	4582	4998
	$(\mu+\lambda)$-CGP-ID (Subgraph)	6037	4069	4863
Multiplier-3Bit	$(1+\lambda)$-CGP	80884	104704	54493
	$(1+\lambda)$-CGP-ID	63651	51526	48538
	$(\mu+\lambda)$-CGP (Subgraph)	70830	60434	45230
Subtractor-2Bit	$(1+\lambda)$-CGP	14216	14347	10657
	$(1+\lambda)$-CGP-ID	**10352**	**9240**	**7728**
	$(\mu+\lambda)$-CGP (Subgraph)	13525	15077	8474

Active Function Node Range

Problem	Algorithm	Mean Active Function Node Range	Median
Parity-Even-3	$(1+\lambda)$-CGP	130	131
	$(1+\lambda)$-CGP-ID	**140**	**147**
Parity-Even-4	$(1+\lambda)$-CGP	139	139
	$(1+\lambda)$-CGP-ID	**160**	**161**
Parity-Even-5	$(1+\lambda)$-CGP	173	174
	$(1+\lambda)$-CGP-ID	**216**	**214**
Parity-Even-6	$(1+\lambda)$-CGP	189	188
	$(1+\lambda)$-CGP-ID	**227**	**232**
Parity-Even-7	$(1+\lambda)$-CGP	163	159
	$(1+\lambda)$-CGP-ID	**195**	**192**
Parity-Even-8	$(1+\lambda)$-CGP	165	164
	$(1+\lambda)$-CGP-ID	**191**	**190**

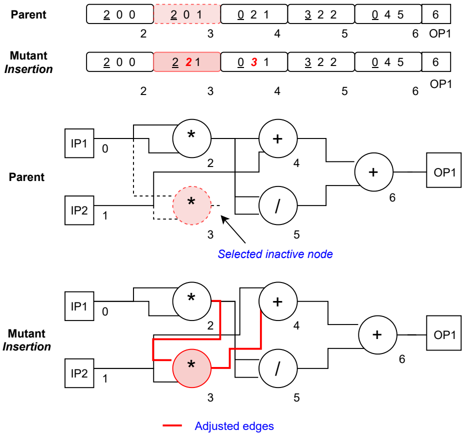
Active Function Node Range

Problem	Algorithm	Mean Active Function Node Range	Median
Adder-1Bit	$(1+\lambda)$-CGP	43	43
	$(1+\lambda)$-CGP-ID	**48**	**48**
Adder-2Bit	$(1+\lambda)$-CGP	48	47
	$(1+\lambda)$-CGP-ID	**77**	**78**
Adder-3Bit	$(1+\lambda)$-CGP	153	152
	$(1+\lambda)$-CGP-ID	**159**	**158**
Multiplier-2Bit	$(1+\lambda)$-CGP	136	136
	$(1+\lambda)$-CGP-ID	**139**	**138**
Multiplier-3Bit	$(1+\lambda)$-CGP	161	159
	$(1+\lambda)$-CGP-ID	**165**	**164**
Subtractor-2Bit	$(1+\lambda)$-CGP	163	159
	$(1+\lambda)$-CGP-ID	**173**	**173**


Parity-Even-3
Parity-Even-4
Parity-Even-5
Parity-Even-6
Parity-Even-7
Parity-Even-8
Mean Active Function Nodes
Generation number
Algorithm
(1+1)-CGP
(1+1)-CGP-ID


Adder-1Bit
Adder-2Bit
Adder-3Bit
Multiplier-2Bit
Multiplier-3Bit
Subtractor-2Bit
Mean Active Function Nodes
Mean Active Function Nodes
Generation number
Generation number
Generation number
Algorithm (1+1)-CGP (1+1)-CGP-ID
Algorithm (1+1)-CGP (1+1)-CGP-ID
Algorithm (1+1)-CGP (1+1)-CGP-ID

Insertion Mutation



Deletion Mutation