THE UNIVERSITY of ADELAIDE

Markus Wagner and Frank Neumann

Single- and Multi-Objective Genetic Programming: New Runtime Results for SORTING

Overview

Genetic Programming (GP):

- Highly complex GP variants address challenging problems, e.g., in symbolic regression
- Currently, it seems to be impossible to analyse these complex variants on complex problems.

Our key questions

- Which optimisation problems can provably be solved by (simple) GPs in polynomial time?
- Can we provide design support to a practitioner?

Current Status "EA Theory"

Computational Complexity Analysis of Evolutionary Computing

- EAs for discrete combinatorial optimisation (lots of results)
- Evolutionary Multi-Objective Optimisation (many results)
- Ant Colony Optimisation (some results)
- EAs for continuous optimisation (initial results)
- Particle Swarm Optimisation (initial results)

- Our Goal: Rigorous insights into the working principles of GP using existing approaches!

Current Status "GP Theory"

Initial article [Durrett/Neumann/O'Reilly 2011]
"GP Computational Complexity on ORDER/MAJORITY"

Properties of the functions:

- Separable (subproblems can be optimised independently)
- Admit multiple solutions

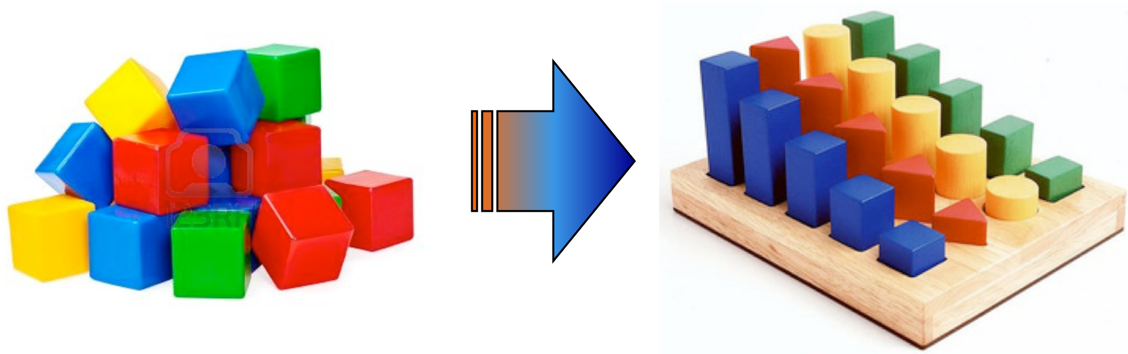
Additional works by Kötzing, Neumann, Nguyen, O'Reilly, Sutton, Urli, and Wagner (2011-2014):

- MAX problem, generalised ORDER/MAJORITY
- Different mutation strategies
- Different multi-objective GPs

In summary:

- Techniques: fitness-based partitions, random walks, coupon collector arguments, drift analysis, failure events, ...
- many bounds known

SORTING



- One of the basic problems in computer science.
- Optimisation problem: maximise the sortedness in a given permutation of elements.
- First combinatorial optimisation problem analysed for EAs.
- Many measures of sortedness work provably well for permutation based EAs (Scharnow/Tinnefeld/Wegener 2002).

http://materikuliah.info/files/2010/11/750403_Sortierbrett_Sorting_Board_01_medium.jpg
http://us.123rf.com/400wm/400/400/pzaxe/pzaxe1104/pzaxe110400104/9402245-a-small-handful-of-children-s-toy-blocks-on-a-white-background.jpg

Measures of Sortedness

Given a permutation s (e.g. 1 3 2 4 5)

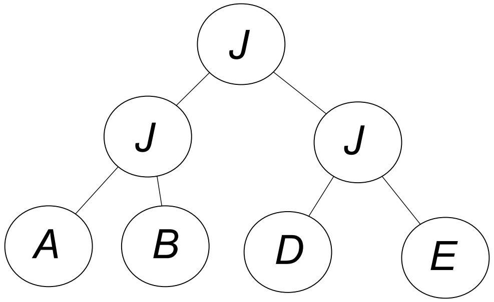
- INV(s) pairs in order in s
- HAM(s) Hamming distance to *optimum*
- RUN(s) number of ascending (sorted) subsequences
- LAS(s) longest ascending sequence length
- EXC(s) number of pairwise exchanges

Scharnow/Tinnefeld/Wegener 2002: Polynomial upper bounds for all functions, except RUN.

GP and SORTING

Four Algorithms

- Tree-based approaches
- Inorder parse leads to (incomplete) permutation**
- Consider different sortedness (fitness) measures



Algorithms (summary)

(1+1)-GP*, F(X)

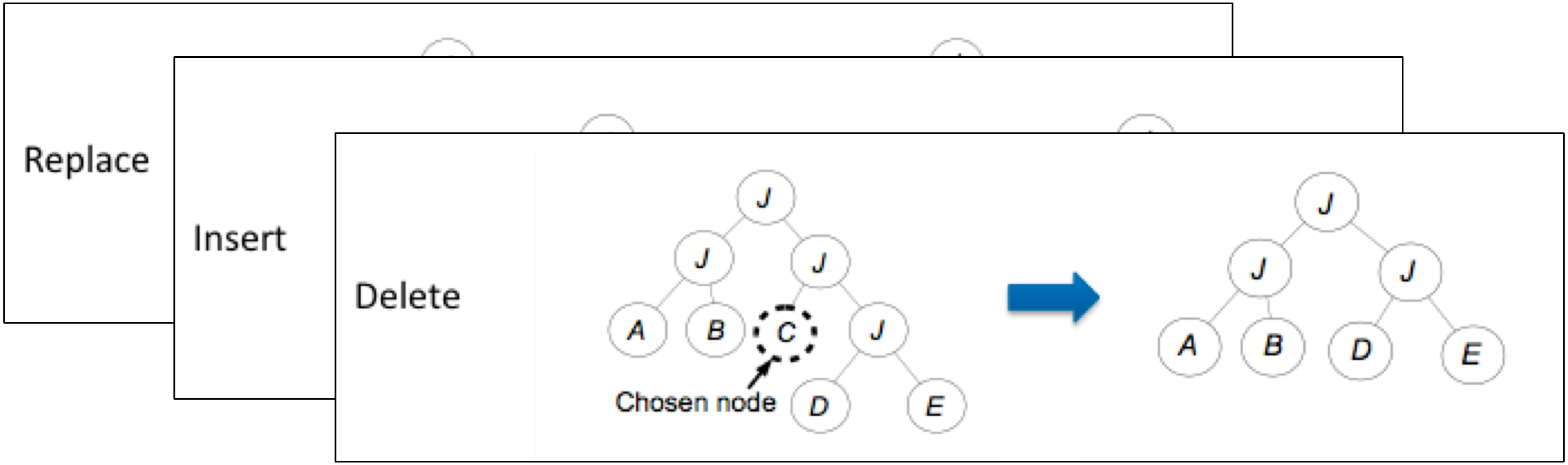
(1+1)-GP, F(X)
- requires: not worse
- noteworthy: no bloat control

(1+1)-GP, MO-F(X)
- requires: at least not longer
- noteworthy: parsimony pressure towards shorter solutions

SMO-GP, MO-F(X)
- requires: weak dominance
- noteworthy: number of different sortedness values limits population size

Variation Operator: HVL-mutate

With equal probability, do…



Choice of parameter k:

- k=1 do a single operation
- k=1+Poisson(1) do multiple operations

Results (before this paper)

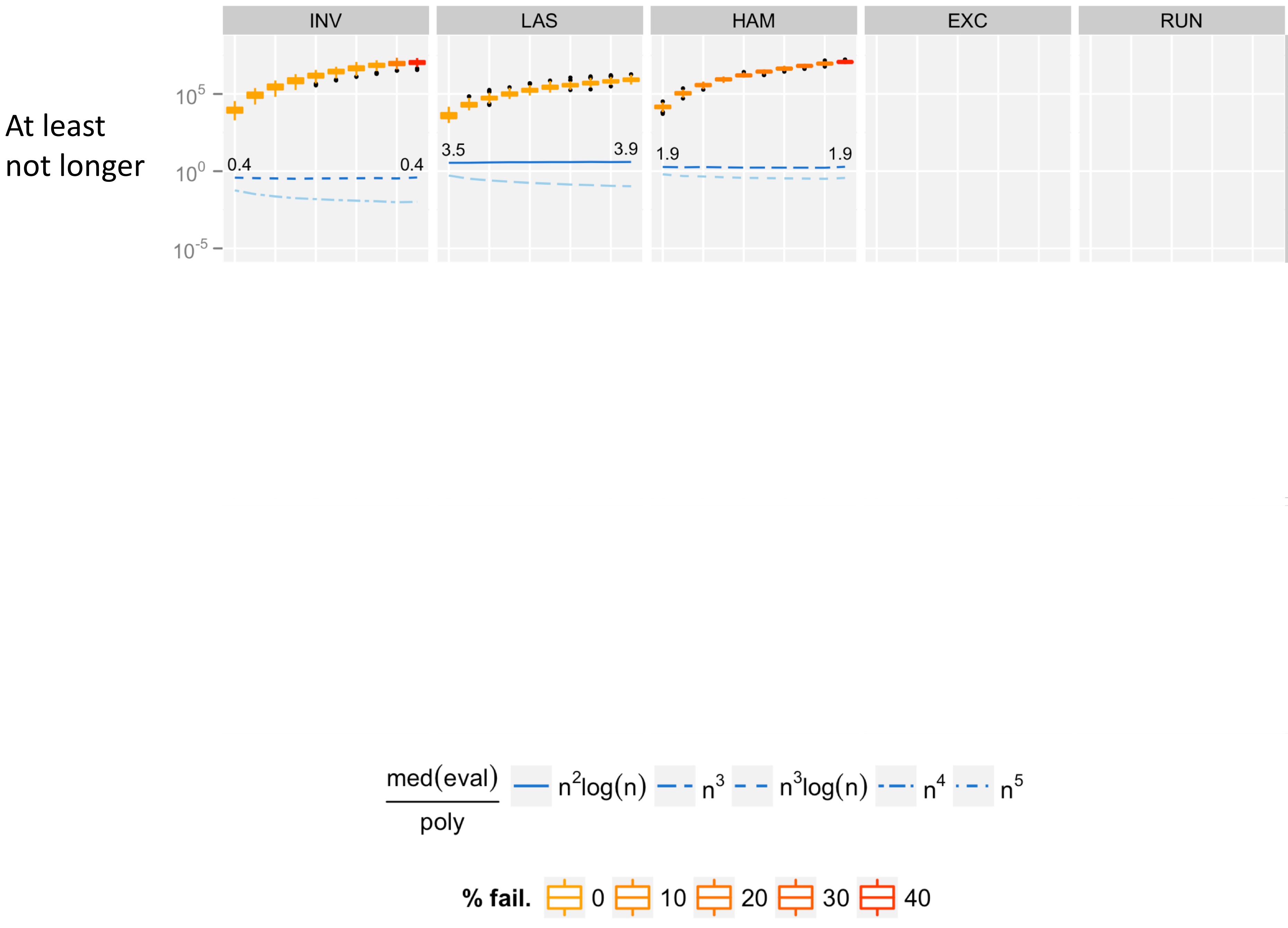
F(X)	(1+1)-GP*, F(X)		(1+1)-GP, F(X)
	single	multi	single/multi
INV			
LAS			
HAM			
EXC			
RUN			

F(X)	(1+1)-GP, MO-F(X)		SMO-GP, MO-F(X)
	single	multi	single/multi
INV			
LAS			
HAM	∞		$O(nT_{init} + n^4)$
EXC	∞		$O(nT_{init} + n^3 \log n)$
RUN	∞		$O(nT_{init} + n^3 \log n)$

Results (*this paper)

F(X)	(1+1)-GP*, F(X) single	(1+1)-GP*, F(X) multi	(1+1)-GP, F(X) single/multi
INV	$O(n^3 T_{max})$ $\star$	$O(n^3 T_{max})$ $\star$	?
LAS	∞ $\star$	$\Omega\left(\left(\frac{n}{e}\right)^n\right)^\star$	
HAM	∞ $\star$	$\Omega\left(\left(\frac{n}{e}\right)^n\right)^\star$	
EXC	∞ $\star$	$\Omega\left(\left(\frac{n}{e}\right)^n\right)^\star$	
RUN	∞ $\star$	$\Omega\left(\left(\frac{n}{e}\right)^n\right)^\star$	

F(X)	(1+1)-GP, MO-F(X) single	(1+1)-GP, MO-F(X) multi	SMO-GP, MO-F(X) single/multi
INV	$O(T_{init} + n^5)$ $\star$	?	[illegible]
LAS	$O(T_{init} + n^2 \log n)$ $\star$	$O(T_{init} + n^2 \log n)$ †*	[illegible]
HAM	∞	?	[illegible]
EXC	∞	?	[illegible]
RUN	∞	?	[illegible]



At least
not longer
INV
LAS
HAM
EXC
RUN
10^5
10^0
10^-5
0.4
0.4
3.5
3.9
1.9
1.9
med(eval)
poly
n^2log(n)
n^3
n^3log(n)
n^4
n^5
% fail.
0
10
20
30
40

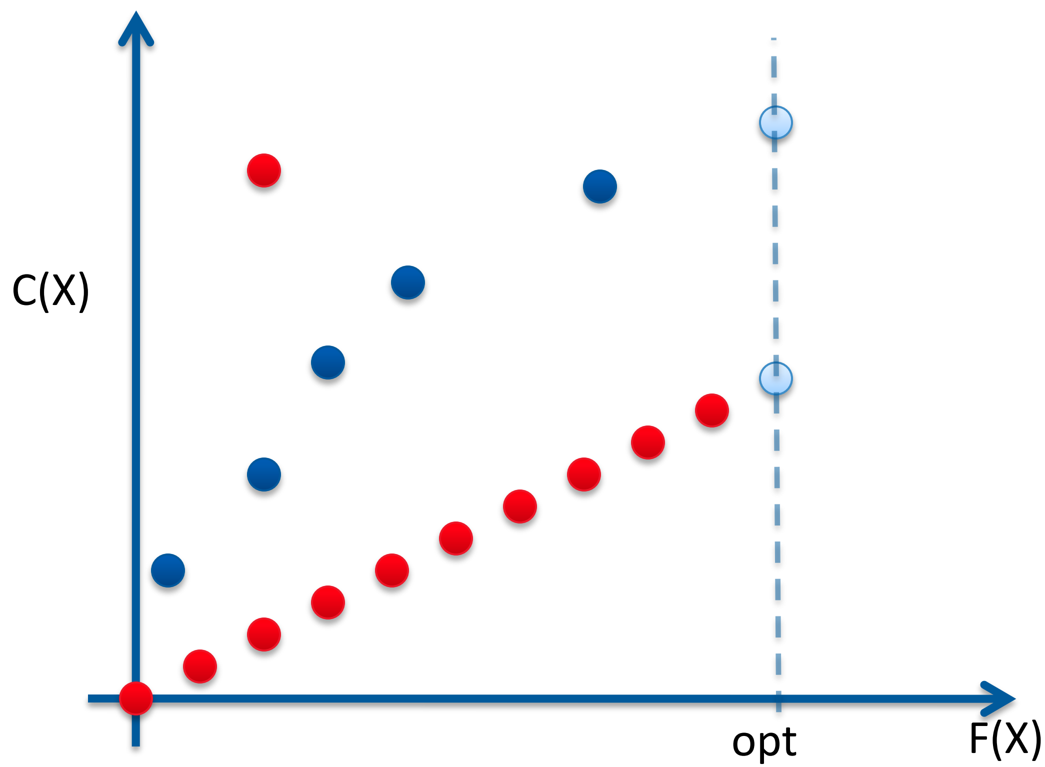
Algorithms (summary)

(1+1)-GP*, F(X)	number of sortedness improving steps limits solution size
(1+1)-GP, F(X)	no bloat control
(1+1)-GP, MO-F(X)	parsimony pressure
SMO-GP, MO-F(X)	number of different sortedness values limits population size

Results SMO-GP

Proof idea:

1. Introduce the empty solution in $O(kT_{init})$
2. Build up the Pareto front step by step.



Polynomial bounds for SMO-GP–single/-multi using INV & LAS

Algorithm (1/4)
(1+1)-GP*-single for maximisation

```
Choose an initial solution X;
repeat
    Set Y := X;
    Apply the mutation operator with k = 1 to Y;   HVL mutate
    if f(Y) > f(X) then set X := Y;
```

Algorithm (1/4)
(1+1)-GP*-single for maximisation

Choose an initial solution X;
repeat
    Set $Y := X$;
    Apply the mutation operator with $k = 1$ to Y;
    if $f(Y) > f(X)$ **then** set $X := Y$;

Algorithm (2/4)
(1+1)-GP -single for maximisation

Choose an initial solution X;
repeat
- Set $Y := X$;
- Apply the mutation operator with $k = 1$ to Y;
- **if** $f(Y) \geqslant f(X)$ **then** set $X := Y$;

Algorithm (3/4)
(1+1)-GP -single for maximisation

Choose an initial solution X;
repeat
    Set $Y := X$;
    Apply the mutation operator with $k = 1$ to Y;
    if $f(Y) \geq f(X)$ **then** set $X := Y$;

Parsimony pressure to favour short solutions: use MO-F(X) instead of F(X)

MO-F(Y) ≥ MO-F(X) holds iff F(Y) > F(X) or (F(Y) = F(X) and C(Y) ≤ C(X))

Algorithm (4/4)
SMO-GP

Choose an initial solution X;
Set $P := \{X\}$;
repeat
    Choose $X \in P$ uniformly at random;
    Set $Y := X$;
    Apply mutation to Y;
    if $\{Z \in P \mid Z \succeq Y\} = \emptyset$ **then** set $P := (P \setminus \{Z \in P \mid Z \succ Y\}) \cup \{Y\}$;

A proper MO algorithm for the sortedness F(X) and the solution quality C(X).

Results (1+1)-GP*

➔ The expected optimisation time is $O(n^3T_{max})$ using INV.

Proof based on fitness-based partition:

- n(n-1)/2+1 different sortedness values possible
- Probability to make an improving mutation $$\frac{1}{3}\cdot\frac{1}{2}\cdot\frac{1}{n}\cdot\frac{1}{T_{max}}=\Omega\left(\frac{1}{nT_{max}}\right)$$
- Overall optimisation time bounded by $$\sum_{k=0}^{n\cdot(n-1)/2} O\left(nT_{max}\right)=O(n^3T_{max})$$

For HAM, LAS, RUN & EXC: local optima exist that can only be left in expected exponential time with n mutations.

Results (1+1)-GP

➔ No results for the (1+1)-GP, F(X).

➔ The expected optimisation time of (1+1)-GP-single on MO-LAS is $O(T_{init}+n^2 \log n)$.

Proof idea:

- Deleting all blocking and surplus leaves takes $O(T_{init}+n \log n)$
- Correctly inserting the missing leaves then takes $O(n^2 \log n)$

"Multi" case: a sortedness improvement may be accompanied by the insertion of many elements…

Results (1+1)-GP

Bound the solution size [t=poly(n) steps and $C(T_{init})$=poly(n)]

- Failure probability for inserting at most n^{ε} in a single HVL operation is $e^{-\Omega(n\varepsilon)}$.
- For LAS and EXC, at most n sortnedness improving steps are possible.
- Thus, the failure probability for adding at most nn^{ε} in t time steps is $te^{-\Omega(n\varepsilon)}=e^{-\Omega(n\varepsilon)}$.
- Thus, the size does not exceed $T_{init}+nn^{\varepsilon}$ within poly(t) time steps, with high probability.

➔ The optimisation time of (1+1)-GP-multi on MO-LAS is $O(T_{init}+n^2 \log n)$, with probability 1-o(1).

Proof idea:

- As before
- Use Chernoff bounds and multiplicative drift with tail bounds to consider multiple mutations.

Methods

Huge set of methods for the analysis is available:

- Fitness-based partitions
- Expected distance decrease
- Coupon Collector's Theorem
- Markov, Chebyshev, Chernoff, Hoeffding bounds
- Markov chain theory: waiting times, first hitting times
- Rapidly mixing Markov chains
- Random walks: gambler's ruin, drift analysis, martingale theory
- Identifying typical events and failure events
- Potential functions

Computational Complexity Analysis

Black Box Scenario

- Measure the runtime T by the number of fitness evaluations.
- Consider time to reach
  - an optimal solution
  - a good approximation

Alternative: Analyse

- expected number of fitness evaluations
- success probability after a fixed number of t steps.

Introduction

There are many

- successful applications and
- experimental studies

of Genetic Programming.

We want to

- argue in a rigorous way about GP algorithms and
- contribute to their theoretical understanding.

This is also important for the acceptance of GP outside the EC community.

Classical Algorithm Analysis

- Classical algorithm analysis has a large focus on runtime and approximation behavior of algorithms.

Our key questions

- Which optimization problems can provably be solved by (simple) GPs in polynomial time?
- (Which functions can provably be learned by (simple) GP systems in polynomial time?)