Machine learning for analytic calculations in theoretical physics

Matthias Wilhelm, University of Southern Denmark

SDU

HAMLET Physics 2025
August 20th, 2025

[2502.05121] with M. von Hippel

see also [2502.09544] by Song, Yang, Cao, Luo, Zhu

see also [2504.16045] by M. Zeng

work in progress with J. Berman, F. Charton and M. Zeng

DANMARKS FRIE FORSKNINGSFOND   QM Centre for Quantum Mathematics   ℏQTC   VILLUM FONDEN

Table of contents

ML for analytic calculations in theoretical physics

Typical machine learning applications: Noisy numeric real-world data

Theoretical physics: Exact analytic calculations

ML for analytic calculations in theoretical physics

Typical machine learning applications: Noisy numeric real-world data

Theoretical physics: Exact analytic calculations

Solutions hard to calculate but easy to check
$\Rightarrow$ Case for **Machine Learning**

ML for analytic calculations in theoretical physics

Typical machine learning applications: Noisy numeric real-world data

Theoretical physics: Exact analytic calculations

Solutions hard to calculate but easy to check
$\Rightarrow$ Case for **Machine Learning**

Examples

- Symbol bootstrap
  [Cai, Charton, Cranmer, Dixon, Merz, Nolte, **MW** (2024)]
  $\rightarrow$ My talk at HAMLET Physics 2024

- Spinor-helicity simplifications [Cheung, Dersy, Schwartz (2024)]
- Integration-by-parts reduction
  [Hippel, **MW** (2025)], [Song, Yang, Cao, Luo, Zhu (2025)], [Zeng (2025)] $\rightarrow$ this talk

The problem in a nutshell

Alternative title: ML for Linear Algebra

The problem in a nutshell

Alternative title: ML for Linear Algebra

Why? Isn't ML harder than Linear Algebra?

The problem in a nutshell

Alternative title: ML for Linear Algebra

Why? Isn't ML harder than Linear Algebra?

Problem: Given a large set S of redundant linear equations, pick a small subset $s \subset S$ that still allows to uniquely solve for a given set of unknowns

The problem in a nutshell

Alternative title: ML for Linear Algebra

Why? Isn't ML harder than Linear Algebra?

Problem: Given a large set S of redundant linear equations, pick a small subset $s \subset S$ that still allows to uniquely solve for a given set of unknowns

Challenge: $|\{s \subset S\}| = 2^{|S|}$ $\rightarrow$ Heuristics $\rightarrow$ ML

Table of contents

Fundamental Physics

Aim: Understand fundamental constituents of **matter** & **interactions**!

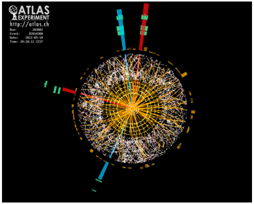


Image: ATLAS

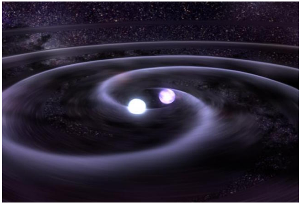


Image: NASA/Goddard Space Flight Center

New experiments $\Rightarrow$ Need for high-precision theory predictions!

Fundamental Physics

Aim: Understand fundamental constituents of **matter** & **interactions**!

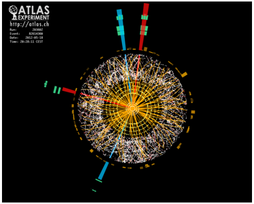


Image: ATLAS

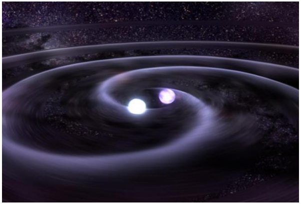
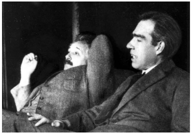


Image: NASA/Goddard Space Flight Center

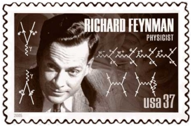
New experiments $\Rightarrow$ Need for high-precision theory predictions!

Theoretical framework:
Quantum Field Theory $=$ Special relativity $+$ Quantum Mechanics



Pipeline for calculating precision predictions

Feynman diagrams → Feynman integrals → Master integrals → Predictions

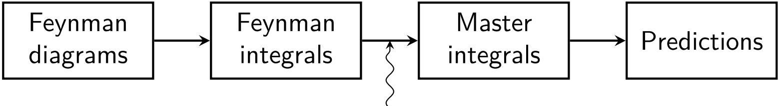


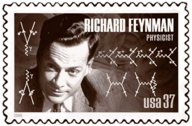
Feynman integrals: $\sim 10^5$

IBP reduction

Master integrals: $\sim 10^2$

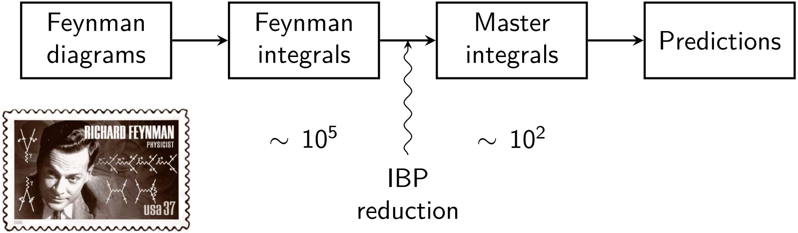
Pipeline for calculating precision predictions





What is integration-by-parts (IBP) reduction?

Pipeline for calculating precision predictions



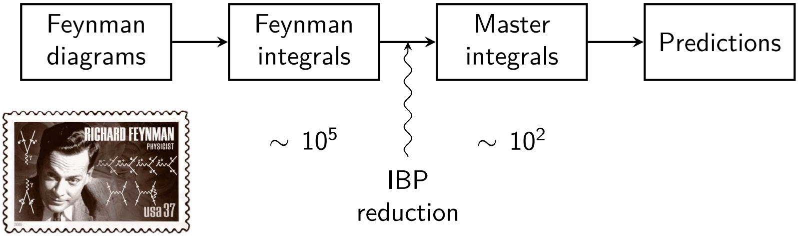
What is integration-by-parts (IBP) reduction?

$\langle$Feynman integrals$\rangle$ = vector space of finite dimension

[Smirnov, Petukhov (2010)]

$\Rightarrow \exists$ finite basis $\{I_1, \ldots, I_N\}$ a.k.a. master integrals

Pipeline for calculating precision predictions



What is integration-by-parts (IBP) reduction?

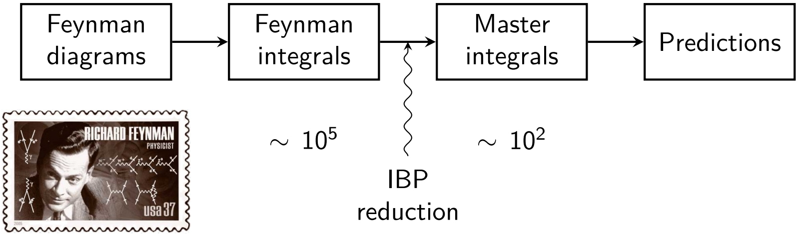
⟨Feynman integrals⟩ = vector space of finite dimension

[Smirnov, Petukhov (2010)]

$\Rightarrow$ $\exists$ finite basis $\{I_1, \ldots, I_N\}$ a.k.a. master integrals

$\Rightarrow$ Decompose $I = \sum_{i=1}^{N} c_i I_i$

Pipeline for calculating precision predictions



What is integration-by-parts (IBP) reduction?

$\langle$Feynman integrals$\rangle$ = vector space of finite dimension

[Smirnov, Petukhov (2010)]

$\Rightarrow$ $\exists$ finite basis $\{I_1, \ldots, I_N\}$ a.k.a. master integrals

$\Rightarrow$ Decompose $I = \sum_{i=1}^{N} c_i I_i$

Bottle neck of many calculations!

e.g. 300k CPU hours [Driesse, Jakobsen, Mogull, Plefka, Sauer, Usovitsch (2024)]

Feynman integrals and IBP identities

General family of Feynman integrals

$$I_{a_1,\ldots,a_n} = \int \frac{\prod_{l=1}^{L} d^D k_l}{\prod_{i=1}^{n} [D_i(k_1^\mu, \ldots, k_L^\mu)]^{a_i}}$$

where $a_i \in \mathbb{Z}$, $L \in \mathbb{N}$, $k_l \in \mathbb{R}^D$, $\mu = 1, \ldots, D$ and D_i polynomials

Feynman integrals and IBP identities

General family of Feynman integrals

$$I_{a_1,\dots,a_n} = \int \frac{\prod_{l=1}^{L} d^D k_l}{\prod_{i=1}^{n} [D_i(k_1^\mu,\dots,k_L^\mu)]^{a_i}}$$

where $a_i \in \mathbb{Z}$, $L \in \mathbb{N}$, $k_l \in \mathbb{R}^D$, $\mu = 1,\dots,D$ and D_i polynomials

Integration-by-part identities [Chetyrkin, Tkachov (1981)]

$$0 = \int \prod_{i=1}^{L} d^D k_i \sum_{\mu=1}^{D} \frac{d}{d k_l^\mu} \frac{q^\mu}{\prod_{i=1}^{n} D_i{}^{a_i}} = \text{linear combination of } I_{a'_1,\dots a'_n}$$

for any $q \in \mathbb{R}^D$ with $a'_i = a_i, a_i \pm 1$

Feynman integrals and IBP identities

General family of Feynman integrals

$$I_{a_1,\ldots,a_n} = \int \frac{\prod_{l=1}^{L} d^D k_l}{\prod_{i=1}^{n} [D_i(k_1^\mu, \ldots, k_L^\mu)]^{a_i}}$$

where $a_i \in \mathbb{Z}$, $L \in \mathbb{N}$, $k_l \in \mathbb{R}^D$, $\mu = 1, \ldots, D$ and D_i polynomials

Integration-by-part identities [Chetyrkin, Tkachov (1981)]

$$0 = \int \prod_{i=1}^{L} d^D k_i \sum_{\mu=1}^{D} \frac{d}{d k_l^\mu} \frac{q^\mu}{\prod_{i=1}^{n} D_i{}^{a_i}} = \text{linear combination of } I_{a'_1,\ldots a'_n}$$

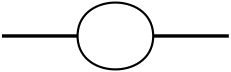
for any $q \in \mathbb{R}^D$ with $a'_i = a_i, a_i \pm 1$

Integration-by-part identities can't be solved for generic a_i
$\rightarrow$ specify to particular values $S \subset \mathbb{Z}^n$ and solve via Laporta's algorithm

[Laporta (2000)]

A simple example

Integral family example: Bubble integral

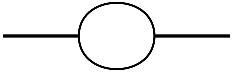


$$I_{a_1,a_2} = \int \frac{d^D k}{(k^2 - m^2)^{a_1} \left[(p-k)^2\right]^{a_2}}$$

with $p \in \mathbb{R}^D$, $m \in \mathbb{R}$

A simple example

Integral family example: Bubble integral



$$I_{a_1,a_2} = \int \frac{d^D k}{(k^2 - m^2)^{a_1} \left[(p-k)^2\right]^{a_2}}$$

with $p \in \mathbb{R}^D$, $m \in \mathbb{R}$

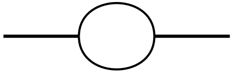
Integration-by-part identities for $q = p, k$:

$$0 = (D-2a_1-a_2) I_{a_1,a_2} - 2a_1 m^2 I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1}$$

$$0 = (a_2-a_1) I_{a_1,a_2} - a_1(m^2+p^2) I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1} + a_1 I_{a_1+1,a_2-1}$$

A simple example

Integral family example: Bubble integral



$$I_{a_1,a_2} = \int \frac{d^D k}{(k^2 - m^2)^{a_1} \left[(p-k)^2\right]^{a_2}}$$

with $p \in \mathbb{R}^D$, $m \in \mathbb{R}$

Integration-by-part identities for $q = p, k$:

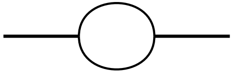
$$0 = (D-2a_1-a_2) I_{a_1,a_2} - 2a_1 m^2 I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1}$$

$$0 = (a_2-a_1) I_{a_1,a_2} - a_1(m^2+p^2) I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1} + a_1 I_{a_1+1,a_2-1}$$

Possible master integrals $I_{1,1}$, $I_{2,0}$

A simple example

Integral family example: Bubble integral



$$I_{a_1,a_2} = \int \frac{d^D k}{(k^2 - m^2)^{a_1} [(p-k)^2]^{a_2}}$$

with $p \in \mathbb{R}^D$, $m \in \mathbb{R}$

Integration-by-part identities for $q = p, k$:

$$0 = (D-2a_1-a_2) I_{a_1,a_2} - 2a_1 m^2 I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1}$$

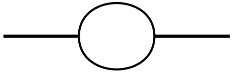
$$0 = (a_2-a_1) I_{a_1,a_2} - a_1(m^2+p^2) I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1} + a_1 I_{a_1+1,a_2-1}$$

Possible master integrals $I_{1,1}, I_{2,0}$

Task: IBP reduce $I_{2,1}$

A simple example

Integral family example: Bubble integral



$$I_{a_1,a_2} = \int \frac{d^D k}{(k^2 - m^2)^{a_1} \left[(p-k)^2\right]^{a_2}}$$

with $p \in \mathbb{R}^D$, $m \in \mathbb{R}$

Integration-by-part identities for $q = p, k$:

$$0 = (D-2a_1-a_2) I_{a_1,a_2} - 2a_1 m^2 I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1}$$

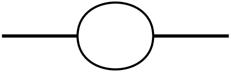
$$0 = (a_2-a_1) I_{a_1,a_2} - a_1(m^2+p^2) I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1} + a_1 I_{a_1+1,a_2-1}$$

Possible master integrals $I_{1,1}, I_{2,0}$

Task: IBP reduce $I_{2,1}$

Solution: Picking $a_1 = a_2 = 1$ above $\Rightarrow I_{2,1} = \frac{D-3}{m^2-p^2} I_{1,1} - \frac{1}{m^2-p^2} I_{2,0}$

Integral family example: Bubble integral



$$I_{a_1,a_2} = \int \frac{d^D k}{(k^2 - m^2)^{a_1} \, [(p-k)^2]^{a_2}}$$

with $p \in \mathbb{R}^D$, $m \in \mathbb{R}$

Integration-by-part identities for $q = p, k$:

$$0 = (D-2a_1-a_2) I_{a_1,a_2} - 2a_1 m^2 I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1}$$

$$0 = (a_2-a_1) I_{a_1,a_2} - a_1(m^2+p^2) I_{a_1+1,a_2} - a_2(m^2-p^2) I_{a_1,a_2+1} - a_2 I_{a_1-1,a_2+1} + a_1 I_{a_1+1,a_2-1}$$

Possible master integrals $I_{1,1}, I_{2,0}$

Task: IBP reduce $I_{2,1}$

Solution: Picking $a_1 = a_2 = 1$ above $\Rightarrow$ $I_{2,1} = \frac{D-3}{m^2-p^2} I_{1,1} - \frac{1}{m^2-p^2} I_{2,0}$

Homework: IBP reduce $I_{5,5}$

Seeding strategies

How to choose seeds? Heuristics!

Define for $(a_1, \ldots, a_n) \in \mathbb{Z}^n$:

$$t \equiv \sum_{a_i>0} 1\,, \qquad r \equiv \sum_{a_i>0} a_i\,, \qquad d \equiv r - t\,, \qquad s \equiv -\sum_{a_i<0} a_i$$

Rectangular Seeding: $S_1 = \{(a_1, \ldots, a_n) \in \mathbb{Z}^n | r \leq r_{\max} \wedge s \leq s_{\max}\}$
w/ parameters $r_{\max}, s_{\max}$

Golden Rule: $S_2 = \{(a_1, \ldots, a_n) \in S_1 | d \leq d_{\max}\}$ w/ parameter $d_{\max}$

[Laporta (2000)]

Homework: IBP reduce $I_{5,5} \to S_1 = S_2 = \{(a_1, a_2) \in \mathbb{Z}^2 | r \leq 9 \wedge s \leq 0\}$

Seeding strategies

How to choose seeds? Heuristics!

Define for $(a_1, \ldots, a_n) \in \mathbb{Z}^n$:

$$t \equiv \sum_{a_i>0} 1\,, \qquad r \equiv \sum_{a_i>0} a_i\,, \qquad d \equiv r - t\,, \qquad s \equiv -\sum_{a_i<0} a_i$$

Rectangular Seeding: $S_1 = \{(a_1, \ldots, a_n) \in \mathbb{Z}^n | r \leq r_{\max} \wedge s \leq s_{\max}\}$
w/ parameters $r_{\max}, s_{\max}$

Golden Rule: $S_2 = \{(a_1, \ldots, a_n) \in S_1 | d \leq d_{\max}\}$ w/ parameter $d_{\max}$

[Laporta (2000)]

Homework: IBP reduce $I_{5,5} \to S_1 = S_2 = \{(a_1, a_2) \in \mathbb{Z}^2 | r \leq 9 \wedge s \leq 0\}$

Improved Seeding: $S_3 = \{(a_1, \ldots, a_n) \in S_2 | s \leq t - l + 1\}$ w/ parameter l

$\Rightarrow$ **Order of magnitude improvements** in number of seeds and time!

[Usovitsch (talk in 2023)]

Idea: **Use ML to discover better heuristics**
for picking seeds $s \subset S$

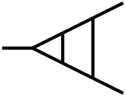
*Always test using  with $a_1, \ldots, a_7$

Table of contents

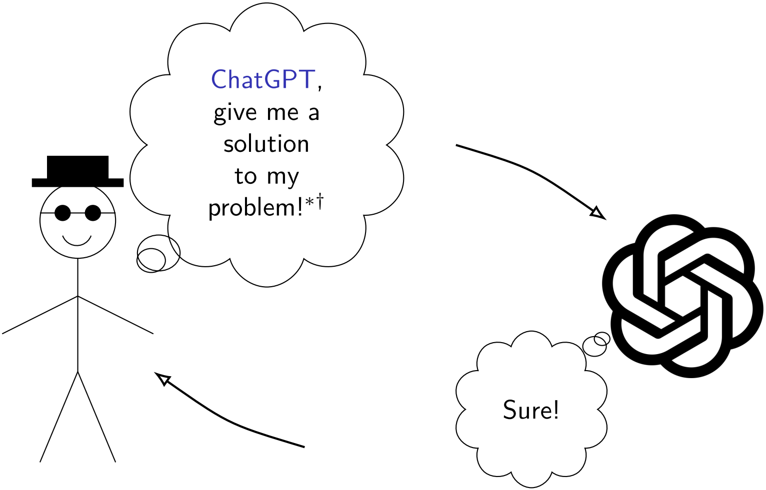
Table of contents

What is funsearch?

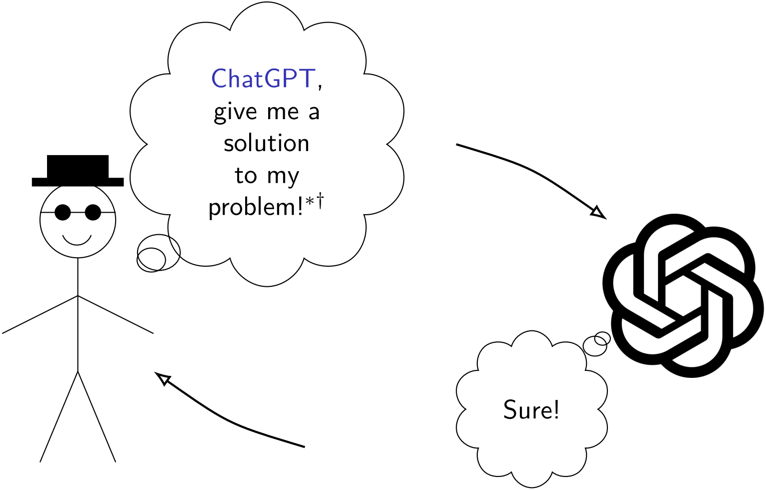
funsearch [Romera-Paredes, Barekatain, Novikov, Balog, Kumar, Dupont, Ruiz, Ellenberg, Wang, Fawzi, Kohli, Fawzi (2023)]

Solving problems by automated brainstorming

Funsearch in a nutshell

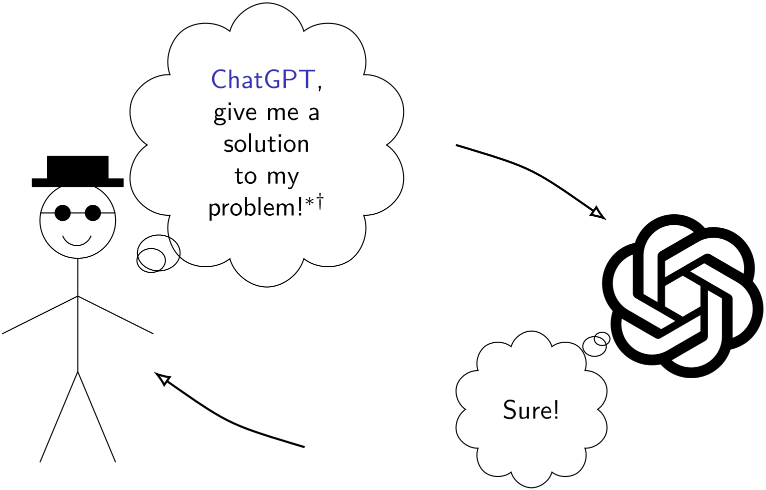


Funsearch in a nutshell



* As python code, so I can check how good it is

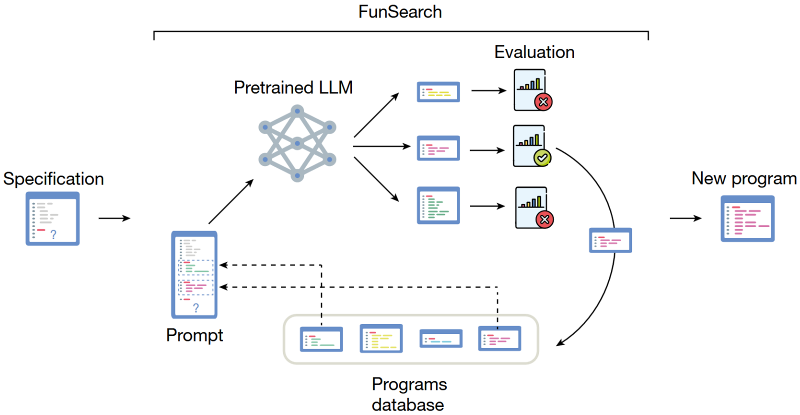
Funsearch in a nutshell



* As python code, so I can check how good it is

† Given those two solutions from earlier already that worked pretty well

Funsearch in slightly longer



[Romera-Paredes, Barekatain, Novikov, Balog, Kumar, Dupont, Ruiz, Ellenberg, Wang, Fawzi, Kohli, Fawzi (2023)]

Properties of funsearch

Some properties:

- Variant of genetic programming, but unconstrained
- Output is python code

$\Rightarrow$ interpretable

$\Rightarrow$ generalizable

[Romera-Paredes, Barekatain, Novikov, Balog, Kumar, Dupont, Ruiz, Ellenberg, Wang, Fawzi, Kohli, Fawzi (2023)]

Properties of funsearch

Some properties:

- Variant of genetic programming, but unconstrained
- Output is python code

$\Rightarrow$ interpretable

$\Rightarrow$ generalizable

[Romera-Paredes, Barekatain, Novikov, Balog, Kumar, Dupont, Ruiz, Ellenberg, Wang, Fawzi, Kohli, Fawzi (2023)]

Fitness

- Gauß elimination $\mathcal{O}(|s|^3) \to$ less seeds is better
- Fitness $= \begin{cases} -|s| & \text{if } \exists \text{ solution} \\ -|s| - |S| & \text{if } \nexists \text{ solution} \end{cases}$

[von Hippel, **MW** (2025)]

Properties of funsearch

Some properties:

- Variant of genetic programming, but unconstrained
- Output is python code

$\Rightarrow$ interpretable

$\Rightarrow$ generalizable

[Romera-Paredes, Barekatain, Novikov, Balog, Kumar, Dupont, Ruiz, Ellenberg, Wang, Fawzi, Kohli, Fawzi (2023)]

Fitness

- Gauß elimination $\mathcal{O}(|s|^3) \to$ less seeds is better
- Fitness $= \begin{cases} -|s| & \text{if } \exists \text{ solution} \\ -|s| - |S| & \text{if } \nexists \text{ solution} \end{cases}$ [von Hippel, **MW** (2025)]
- Sparse $\to$ Refined fitness $= \#$ element-wise operations [Zeng (2025)]

Experiment with funsearch

Starting point corresponding to a golden rule system with $d_{\max} = 1$:

```python
def priority(a_list: list[int]) -> bool:
    """Decides whether to include the seed a_list in the ibp system.
      Returns True or False."""

    len_alist=len(a_list)

    #Number of propagators, which are entries in a_list greater than zero
    num_props=sum(map(lambda x: 1 if x>0 else 0,a_list))

    #Numbers of numerators, which are entries in a_list less than zero
    numerators=sum(map(lambda x: 1 if x<0 else 0,a_list))

    #Dots, the sum of all entries in a_list greater than one
    dots=sum(map(lambda x: x-1 if x>1 else 0,a_list))

    #The simplest choice: if there is more than one dot, exclude the seed
    #else include it
    if dots>1:
      return False
    else:
      return True
```

$\Rightarrow$ 2,148 seeds

[von Hippel, **MW** (2025)]

Experimental results

After 2,400 generation: 2,148 seeds $\rightarrow$ 92 seeds
$\widehat{=}$ improved seeding strategy with $d_{\max} = 0$ and $l = 4$
$\Rightarrow$ **Rediscovered state of the art!**

[von Hippel, **MW** (2025)]

Experimental results

After 2,400 generation: 2,148 seeds $\rightarrow$ 92 seeds
$\hat{=}$ improved seeding strategy with $d_{\max} = 0$ and $l = 4$
$\Rightarrow$ **Rediscovered state of the art!**

After 3,800 generation: 2,148 seeds $\rightarrow$ 88 seeds
$\hat{=}$ improved seeding strategy $+$ $t \geq 4$
$\Rightarrow$ **Improvement on state of the art!**

[von Hippel, **MW** (2025)]

Table of contents

Classic genetic programming

Funsearch: unconstrained + slow → Exploration

Classic genetic programming

Funsearch: unconstrained $+$ slow $\rightarrow$ Exploration

Classic genetic programming: constrained $+$ fast $\rightarrow$ Exploitation
e.g. [Distributed Evolutionary Algorithms in Python (DEAP)]

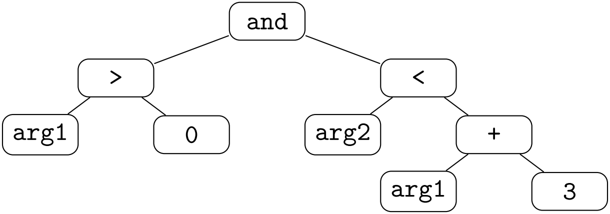
Classic genetic programming

Funsearch: unconstrained + slow → Exploration

Classic genetic programming: constrained + fast → Exploitation
e.g. [Distributed Evolutionary Algorithms in Python (DEAP)]

Syntax trees

```
def func(arg1,arg2):
    return arg1>0 and arg2<arg1+3
```



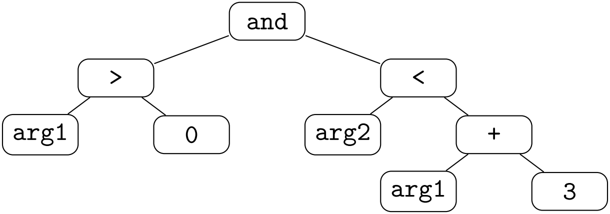
Classic genetic programming

Funsearch: unconstrained + slow → Exploration

Classic genetic programming: constrained + fast → Exploitation
e.g. [Distributed Evolutionary Algorithms in Python (DEAP)]

Syntax trees

```
def func(arg1, arg2):
    return arg1>0 and arg2<arg1+3
```



3+True Nonsense ⇒ **Strongly typed genetic programming**

Strongly typed genetic programming

Evolving trees $=$ grafting


Image: nwtree.com

Strongly typed genetic programming

Evolving trees = grafting

Image: nwtree.com

Cross-over Replace random sub-tree by random sub-tree of another tree
Mutation Replace random sub-tree by random new sub-tree

Strongly typed genetic programming

Evolving trees = grafting


Image: nwtree.com

Cross-over Replace random sub-tree by random sub-tree of another tree
Mutation Replace random sub-tree by random new sub-tree

Building blocks

- Arguments built from $(a_1, \ldots, a_n) \in \mathbb{Z}$: $\sum_{a_i>0} a_i$, $\sum_{a_i>1} a_i$, $-\sum_{a_i<0} a_i$, $\sum_{a_i} a_i$, $\sum_{a_i>0} 1$, $\sum_{a_i>1} 1$, $\sum_{a_i<0} 1$, $\sum_{a_i=0} 1$, $\sum_{a_i=1} 1$, n
- Primitives: `and`, $>$, $<$, $=$, $+$, $-$
- Terminal elements: `True`, $r_{\max}$, $s_{\max}$, $-10, \ldots, +10$

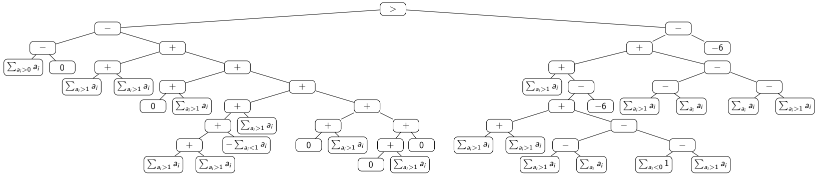
Experimental results

After 18 generation: 2,148 seeds $\rightarrow$ 88 seeds
$\hat{=}$ improved seeding strategy $+$ $t \geq 4$
$\Rightarrow$ **Same result as funsearch but faster!**

[von Hippel, **MW** (2025)]

After 18 generation: 2,148 seeds $\rightarrow$ 88 seeds
$\widehat{=}$ improved seeding strategy $+\ t \geq 4$
$\Rightarrow$ **Same result as funsearch but faster!**

Final tree



[von Hippel, **MW** (2025)]

Table of contents

Conclusion

- **Bottle neck** for analytic calculations in theoretical physics:
  IBP reductions

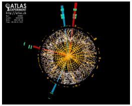


Image: ATLAS

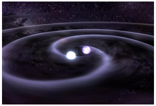
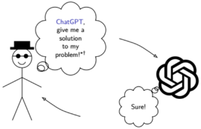
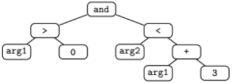


Image: NASA/Goddard Space Flight Center

- **Challenge:** Pick set $s \subset S$ of linear equations
- **ML approach:**
  - funsearch



  - strongly typed genetic programming

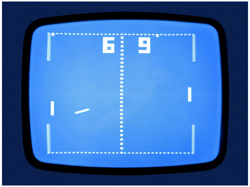


⇒ Improvement over state of the art ⇒ **Proof of principle!**

[von Hippel, **MW** (2025)]

Outlook

- funsearch $\rightarrow$ AlphaEvolve
  [Novikov,Vũ,Eisenberger,Dupont,Huang,Wagner,Shirobokov,Kozlovskii et al. (2025)]
- Gamification $\rightarrow$ RL
  [Zeng (2025)], [Berman, Charton, **MW**, Zeng (in progress)]

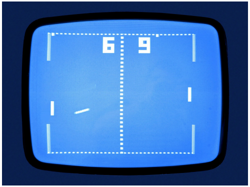


- Generalization & Deployment
  - Interpret $\Rightarrow$ Deploy analytic seeding strategy
  - Finite field techniques $\rightarrow$ Training budget of $10^5 - 10^6$ runs
    $\Rightarrow$ Deploy ML
- ...

Image: Road Travel America

Outlook

- funsearch → AlphaEvolve
  [Novikov,Vũ,Eisenberger,Dupont,Huang,Wagner,Shirobokov,Kozlovskii et al. (2025)]
- Gamification → RL
  [Zeng (2025)], [Berman, Charton, **MW**, Zeng (in progress)]



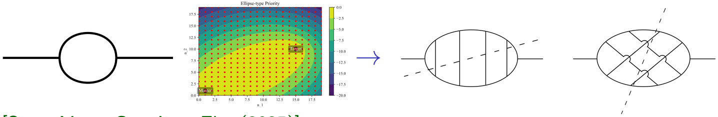
- Generalization & Deployment
  - Interpret ⇒ Deploy analytic seeding strategy
  - Finite field techniques → Training budget of $10^5 - 10^6$ runs
    ⇒ Deploy ML
- ...

Thank you!


Image: Road Travel America

Option A

Interpretable $\Rightarrow$ **Deploy analytic seeding strategy**

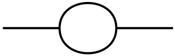
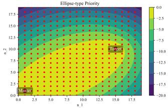
- $t \geq 4$ [von Hippel, **MW** (2025)]
- 

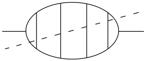
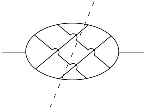
[Song, Yang, Cao, Luo, Zhu (2025)]

Option A

Interpretable $\Rightarrow$ **Deploy analytic seeding strategy**

- $t \geq 4$ [von Hippel, **MW** (2025)]

-  

$\rightarrow$  

  [Song, Yang, Cao, Luo, Zhu (2025)]

Option B

Finite field techniques [von Manteuffel, Schabinger (2014)], [Peraro (2016)]

- Run for $D, m_i, s_{ij} \in \mathbb{F}_p$
- Reconstructs rational dependence on D, m_i, s_{ij}

$\rightarrow$ Training budget of $10^5 - 10^6$ runs $\Rightarrow$ **Deploy ML**