Games and Loop Integrals

Ben Ruijl*, Franz Herzog, Ali Mirsoleimani, Takahiro Ueda, Jos Vermaseren, Andreas Vogt

*LCDS Leiden and Nikhef Amsterdam

Aug 22, 2017

Introduction

Two big problems with higher loop calculations:

- Numerical integration is slow due to huge expressions
- Reducing integrals to simpler ones is very slow

Project HEPGame

Apply successful tools from AI used for games to problems in HEP

Example of a short expression

$$
\begin{aligned}
&+32o^3n^2m + 32o^3n^2l - 48o^3n^2km - 48o^3n^2kl + 32o^4j^2m + 32o^4j^2l + 64o^4ijm - 128o^4ijh + 64o^4ijl + 64o^4i^2m \\
&-128o^4i^2h + 64o^4i^2l - 128o^4gjh + 64o^4gim - 128o^4gih + 64o^4gil + 32o^4g^2m + 32o^4g^2l - 64o^4km - 64o^4kl \\
&+32o^4kjm + 32o^4kjl - 32o^4kj^2m - 32o^4kj^2l + 64o^4kim - 192o^4kih + 64o^4kil - 64o^4kijm + 128o^4kijh - 64o^4kijl \\
&-64o^4ki^2m + 128o^4ki^2h - 64o^4ki^2l + 96o^4kgm - 192o^4kgh + 96o^4kgl + 128o^4kgjh - 64o^4kgim + 128o^4kgih \\
&-64o^4kgil - 32o^4kg^2m - 32o^4kg^2l + 64o^4k^2m + 64o^4k^2l - 64o^4k^2jm - 64o^4k^2jl - 64o^4k^2im - 64o^4k^2il \\
&-64o^4k^2gm - 64o^4k^2gl - 32o^4k^3m - 32o^4k^3l + 48fo^2n^2m + 32fo^2n^2h + 48fo^2n^2l - 48fo^2n^2jm + 64fo^2n^2jh \\
&-48n^3h^2 - 48fo^2n^2jl - 96fo^2n^2im - 96fo^2n^2il - 64fo^2n^2gh - 48fo^2n^2km - 48fo^2n^2kl + 256fo^3jh + 32fo^3j^2m \\
&-128fo^3j^2h + 32fo^3j^2l - 32fo^3j^3m - 32fo^3j^3l - 64fo^3im + 256fo^3ih - 64fo^3il + 128fo^3ijm - 448fo^3ijh \\
&+128fo^3ijl - 128fo^3ij^2m + 64fo^3ij^2h - 128fo^3ij^2l + 192fo^3i^2m - 384fo^3i^2h + 192fo^3i^2l - 192fo^3i^2jm \\
&+256fo^3i^2jh - 192fo^3i^2jl - 128fo^3i^3m + 128fo^3i^3h - 128fo^3i^3l + 64fo^3gm + 64fo^3gl - 448fo^3gjh \\
&+64fo^3gj^2h + 192fo^3gim - 576fo^3gih + 192fo^3gil - 64fo^3gijm + 384fo^3gijh - 64fo^3gijl - 128fo^3gi^2m \\
&+128fo^3gi^2h - 128fo^3gi^2l + 32fo^3g^2m - 64fo^3g^2h + 32fo^3g^2l - 32fo^3g^2jm + 128fo^3g^2jh - 32fo^3g^2jl \\
&-64fo^3g^2im - 64fo^3g^2ih - 64fo^3g^2il - 64fo^3g^3h - 64fo^3km + 128fo^3kh - 64fo^3kl + 32fo^3kjm - 448fo^3kjh \\
&+32fo^3kjl - 96fo^3kj^2m + 256fo^3kj^2h - 96fo^3kj^2l + 128f^2o^2i^2m - 384f^2o^2i^2h + 128f^2o^2i^2l - 384f^2o^2i^2jm
\end{aligned}
$$

Horner schemes

- Say we have $x^3y^2 + x^2y + x^3z$

Horner schemes

- Say we have $x^3y^2 + x^2y + x^3z$
- $x \cdot x \cdot x \cdot y \cdot y + x \cdot x \cdot y + x \cdot x \cdot x \cdot z$
- $9 \times \cdot$

Horner schemes

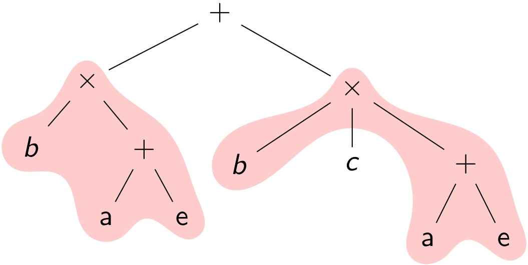
- Say we have $x^3y^2 + x^2y + x^3z$
- $x \cdot x \cdot x \cdot y \cdot y + x \cdot x \cdot y + x \cdot x \cdot x \cdot z$
- $9 \times \cdot$
- Horner scheme: $x^2(y + x(y^2 + z))$
- $x \cdot x \cdot (y + x \cdot (y \cdot y + z))$
- $4 \times \cdot$

Horner schemes

- Say we have $x^3y^2 + x^2y + x^3z$
- $x \cdot x \cdot x \cdot y \cdot y + x \cdot x \cdot y + x \cdot x \cdot x \cdot z$
- $9 \times \cdot$
- Horner scheme: $x^2(y + x(y^2 + z))$
- $x \cdot x \cdot (y + x \cdot (y \cdot y + z))$
- $4 \times \cdot$
- Other possibility: $x^3z + y(x^2(1 + xy))$
- $x \cdot x \cdot x \cdot z + y \cdot (x \cdot x \cdot (1 + x \cdot y))$
- $7 \times \cdot$
- Optimal order problem is NP-hard

Common SubExpression Elimination (CSEE)

- Find most common subexpressions and build a replacement tree
- $b \times (a + e)$ is found
- CSEE reduces both $\times$ and $+$



Research question

- Interplay between Horner and CSEE
- No heuristics available
- Evaluation function is slow (seconds for one sample!)

Question

How do we find the scheme that reduces the number of operations the most with the minimal amount of samples?

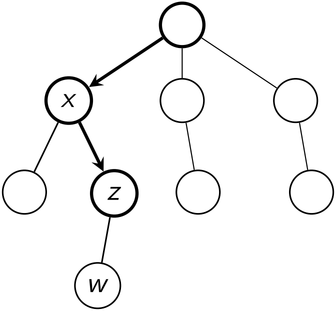
Monte Carlo Tree Search (MCTS)

- Successful for Go (AlphaGo, etc.)
- Build a state tree selectively
- Each node is a variable

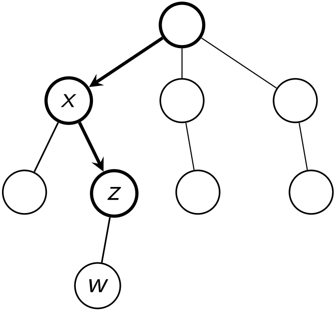
Idea

Use MCTS to find near optimal Horner scheme

Selection



Selection

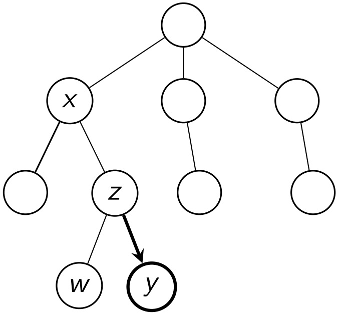


Criterion:

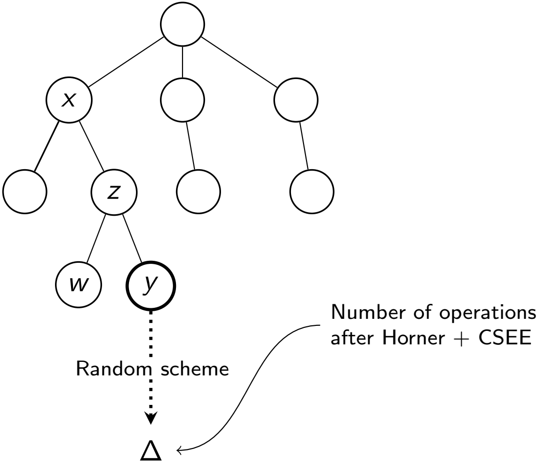
$$\underset{\text{children } c \text{ of } s}{\operatorname{argmax}} \frac{x(c)}{n(c)} + 2C_p\sqrt{\frac{2\ln n(s)}{n(c)}}$$

- $x(c)$ is score of node c
- $n(s)$ is visits at node s
- C_p is exploration-exploitation constant [expensive tuning]

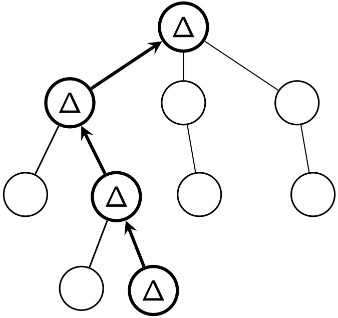
Expansion



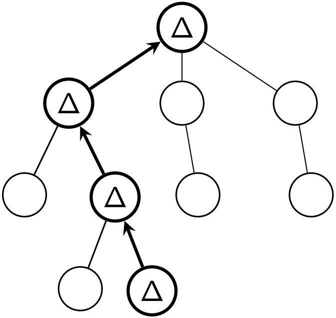
Simulation


x
z
w
y
Random scheme
Δ
Number of operations
after Horner + CSEE

Backpropagation



Backpropagation



MCTS loop:

- Keep on sampling and updating the tree in a best-first way

Downsides:

- Evaluations take a long time for large expressions
- Tuning C_p is hard

Local Stochastic Search

- Investigate the state space
- Define a neighbour as a random swap in Horner scheme:

$$a \quad b \quad c \quad d \quad \longrightarrow \quad c \quad b \quad a \quad d$$

- Move to a random neighbour if it has a better score

Local minima?

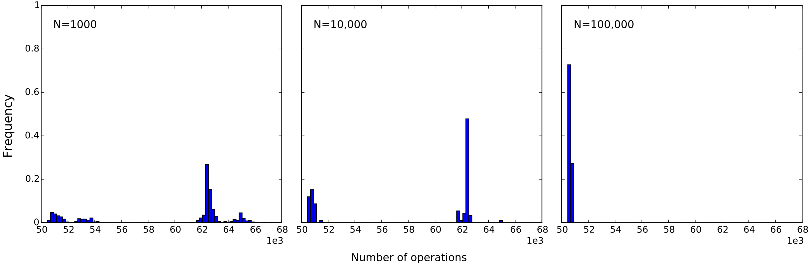


Figure: ‘Local minima’ disappear with more steps ⇒ just saddle points!

Bumpy terrain?

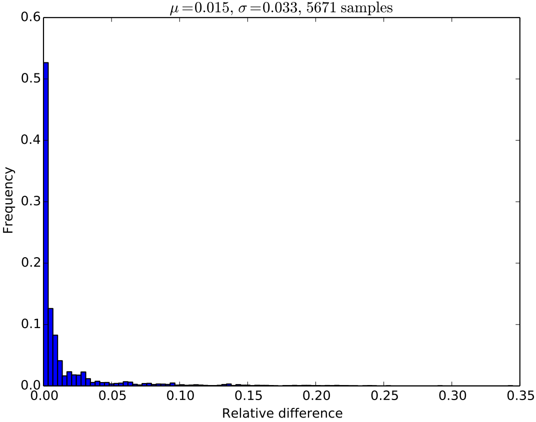


Figure: Most neighbours vary little

Our state space

Figure: A sawa, very flat and plateau-like

Conclusion

- A simple search algorithm without tuning works as well
- No expensive tuning required

Challenges at four and five loops

- Large amount of terms:

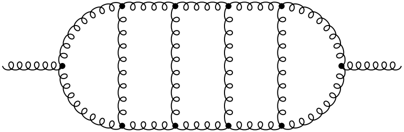
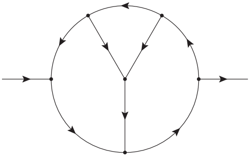


Figure: Represents 12 029 521 scalar integrals!

- Hard to compute integrals
- Mostly slow algorithms

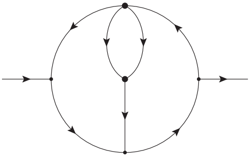
IBP identities

Through integration by parts (IBP) identities we find rules to remove lines:



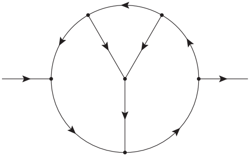
IBP identities

Through integration by parts (IBP) identities we find rules to remove lines:



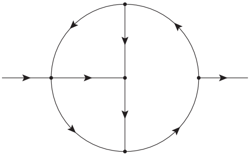
IBP identities

Through integration by parts (IBP) identities we find rules to remove lines:



IBP identities

Through integration by parts (IBP) identities we find rules to remove lines:



3-loop reduction graph

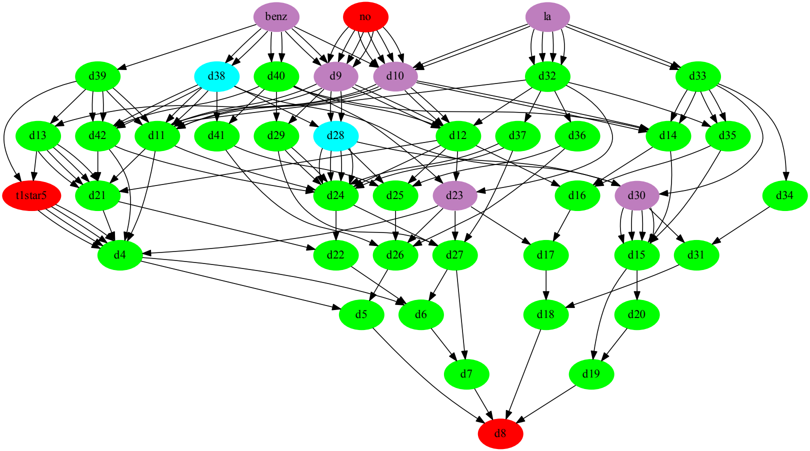


Figure: Automatically generated reduction program for 3-loop diagrams. Each node is a topology, each color a different reduction operation.

4-loop reduction graph

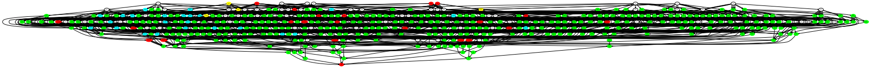


Figure: Reduction graph for 4 loop diagrams

4-loop reduction graph

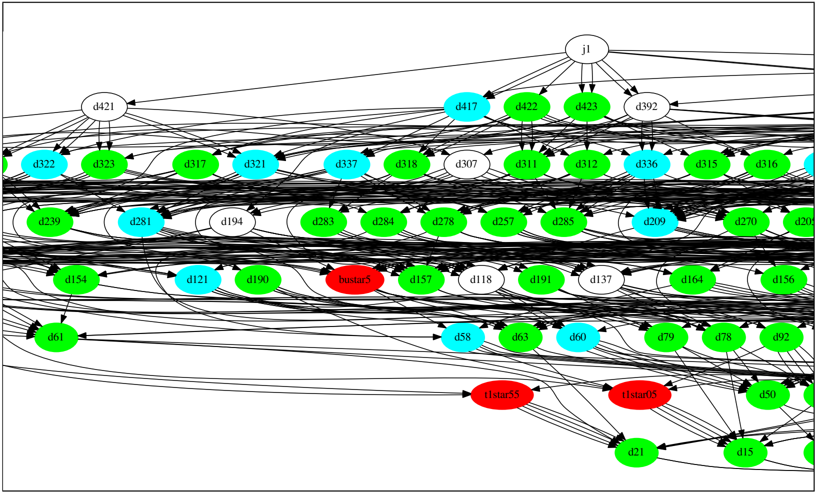
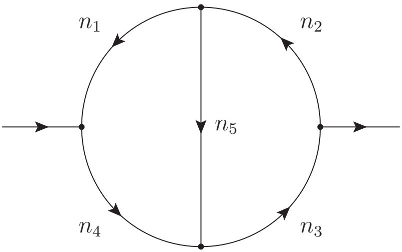


Figure: Part of reduction graph for 4 loops

Solving integration-by-parts identities

- Solve a linear system to express integrals in simpler ones
- Solution: one of the indices is reduced to 0 (or 1 for master integrals)
- Computer-assisted solution
- Possibility for SMT-solvers?



$$I(n_1,n_2,n_3,n_4,n_5) = n_1 I(n_1-1,n_2,n_3,n_4,n_5) + n_2 I(n_1,n_2+1,n_3,n_4,n_5) + \dots$$

$$I(n_1,n_2,n_3,n_4,n_5) = n_3 I(n_1,n_2,n_3-1,n_4,n_5) + n_3 I(n_1,n_2+1,n_3,n_4,n_5) + \dots$$

Reduction rule

```
id,ifmatch->bubu1,
    Z(n1?pos_,n2?pos_,n3?pos_,n4?pos_,n5?pos_,n6?pos_,n7?pos_,
    n8?pos_,n9?pos_,n10?neg0_,n11?neg0_,n12?neg0_,n13?neg0_,n14?neg_)
       = -rat(1,-2*ep-2*n1-n3-n6-n12-n14+4)*(
+Z(-1+n1,-1+n2,n3,n4,1+n5,n6,n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(-n5,1)
+Z(-1+n1,1+n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(n2,1)
+Z(-1+n1,1+n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,1+n14)*rat(-n2,1)
+Z(-1+n1,n2,1+n3,n4,n5,n6,n7,n8,-1+n9,n10,n11,n12,n13,1+n14)*rat(-n3,1)
+Z(-1+n1,n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,1+n12,n13,1+n14)*rat(-n12,1)
+Z(-1+n1,n2,n3,n4,1+n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,1+n14)*rat(n5,1)
+Z(-1+n1,n2,n3,n4,n5,n6,-1+n7,n8,n9,n10,n11,n12,1+n13,1+n14)*rat(-n13,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,1+n12,n13,1+n14)*rat(2*n12,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,2+n14)*rat(2*n14+2,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,n8,-1+n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,n8,n9,1+n10,-1+n11,n12,n13,1+n14)*rat(-n10,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,1+n12,n13,n14)*rat(-n12,1)
+Z(-1+n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(-n2+n5,1)
+Z(n1,-1+n2,-1+n3,n4,n5,n6,n7,n8,n9,n10,n11,1+n12,n13,1+n14)*rat(n12,1)
+Z(n1,-1+n2,-1+n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,2+n14)*rat(1+n14,1)
+Z(n1,-1+n2,1+n3,n4,n5,n6,n7,n8,-1+n9,n10,n11,n12,n13,1+n14)*rat(n3,1)
```

```
+Z(n1,-1+n2,n3,-1+n4,n5,n6,n7,n8,n9,n10,1+n11,n12,n13,1+n14)*rat(n11,1)
+Z(n1,-1+n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,1+n12,n13,1+n14)*rat(n12,1)
+Z(n1,-1+n2,n3,n4,n5,-1+n6,n7,n8,n9,n10,n11,n12,n13,2+n14)*rat(-n14-1,1)
+Z(n1,-1+n2,n3,n4,n5,n6,-1+n7,1+n8,n9,n10,n11,n12,n13,1+n14)*rat(2*n8,1)
+Z(n1,-1+n2,n3,n4,n5,n6,-1+n7,n8,n9,n10,1+n11,n12,n13,1+n14)*rat(-n11,1)
+Z(n1,-1+n2,n3,n4,n5,n6,-1+n7,n8,n9,n10,n11,n12,1+n13,1+n14)*rat(-n13,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,1+n12,n13,1+n14)*rat(-2*n12,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,2+n14)*rat(-2*n14-2,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,1+n8,-1+n9,n10,n11,n12,n13,1+n14)*rat(-n8,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,n8,-1+n9,n10,n11,1+n12,n13,1+n14)*rat(-2*n12,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,n8,n9,1+n10,-1+n11,n12,n13,1+n14)*rat(n10,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,n8,n9,1+n10,n11,n12,n13,1+n14)*rat(-2*n10,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,n8,n9,n10,1+n11,n12,n13,1+n14)*rat(-n11,1)
+Z(n1,-1+n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,1+n12,n13,n14)*rat(-n12,1)
+Z(n1,1+n2,n3,n4,-1+n5,n6,n7,n8,n9,-1+n10,n11,n12,n13,1+n14)*rat(-n2,1)
+Z(n1,1+n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(-n2,1)
+Z(n1,1+n2,n3,n4,n5,n6,n7,-1+n8,n9,-1+n10,n11,n12,n13,1+n14)*rat(2*n2,1)
+Z(n1,1+n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,1+n14)*rat(n2,1)
+Z(n1,n2,-1+n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,1+n12,n13,1+n14)*rat(-n12,1)
+Z(n1,n2,-1+n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,2+n14)*rat(-n14-1,1)
+Z(n1,n2,1+n3,n4,n5,n6,n7,n8,-1+n9,-1+n10,n11,n12,n13,1+n14)*rat(n3,1)
```

```
+Z(n1,n2,n3,n4,-1+n5,n6,-1+n7,n8,n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,-1+n9,n10,n11,n12,1+n13,1+n14)*rat(-n13,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,-1+n9,n10,n11,n12,n13,2+n14)*rat(1+n14,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,n9,-1+n10,n11,1+n12,n13,1+n14)*rat(n12,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,-1+n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,1+n12,n13,1+n14)*rat(n12,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(n1,n2,n3,n4,-1+n5,n6,n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(-n2+n8-n13,1)
+Z(n1,n2,n3,n4,n5,-1+n6,n7,-1+n8,n9,n10,n11,n12,n13,2+n14)*rat(1+n14,1)
+Z(n1,n2,n3,n4,n5,n6,-1+n7,-1+n8,n9,n10,1+n11,n12,n13,1+n14)*rat(n11,1)
+Z(n1,n2,n3,n4,n5,n6,-1+n7,-1+n8,n9,n10,n11,n12,1+n13,1+n14)*rat(-2*n13,1)
+Z(n1,n2,n3,n4,n5,n6,-1+n7,n8,n9,-1+n10,1+n11,n12,n13,1+n14)*rat(n11,1)
+Z(n1,n2,n3,n4,n5,n6,-1+n7,n8,n9,n10,n11,-1+n12,n13,2+n14)*rat(1+n14,1)
+Z(n1,n2,n3,n4,n5,n6,-1+n7,n8,n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(n1,n2,n3,n4,n5,n6,-1+n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(-2*ep-2*n4-1,1)
+Z(n1,n2,n3,n4,n5,n6,1+n7,-1+n8,n9,n10,n11,n12,n13,1+n14)*rat(-n7,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,-1+n9,n10,n11,n12,1+n13,1+n14)*rat(n13,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,-1+n9,n10,n11,n12,n13,2+n14)*rat(-2*n14-2,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,-1+n10,1+n11,n12,n13,1+n14)*rat(-n11,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,-1+n10,n11,1+n12,n13,1+n14)*rat(-2*n12,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,-1+n10,n11,n12,n13,2+n14)*rat(-2*n14-2,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,1+n10,n11,n12,n13,1+n14)*rat(2*n10,1)
```

```
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,-1+n11,n12,1+n13,1+n14)*rat(-2*n13,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,1+n11,n12,n13,1+n14)*rat(n11,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,-1+n12,n13,2+n14)*rat(-n14-1,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,1+n12,n13,1+n14)*rat(-2*n12,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,1+n13,1+n14)*rat(-3*n13,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,1+n14)*rat(10*ep+2*n1,1)
+Z(n1,n2,n3,n4,n5,n6,n7,-1+n8,n9,n10,n11,n12,n13,2+n14)*rat(-2*n14-2,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,-1+n9,-1+n10,n11,1+n12,n13,1+n14)*rat(-n12,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,-1+n9,-1+n10,n11,n12,1+n13,1+n14)*rat(-n13,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,-1+n9,n10,-1+n11,n12,n13,2+n14)*rat(-n14-1,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,-1+n9,n10,n11,n12,1+n13,1+n14)*rat(-n13,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,-1+n9,n10,n11,n12,n13,1+n14)*rat(-4*ep-2*n1-n3,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,n9,-1+n10,n11,n12,n13,1+n14)*rat(-n5+1,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,n9,1+n10,-1+n11,n12,n13,1+n14)*rat(n10,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,-1+n11,n12,n13,1+n14)*rat(2*ep+n5+2*n8
    +n9+n10+n11-5,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,1+n12,n13,n14)*rat(n12,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,-1+n13,1+n14)*rat(-2*ep-n5-2*n8
    -n9-n11-n14+3,1)
+Z(n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,1+n14)*rat(2*ep+n2+n7+
    2*n8+n9+n11+n14-4,1)
);
```

Results at five loops

- Computed five loop beta function for general colour group
  - Took 6 days on a pc with 32 cores
- Recomputed $H \to b\bar{b}$, R-ratio
  - Easy: took a few hours on one pc
- Computed $H \to gg$
  - Quartically divergent diagrams...
  - Hard: took two months

Acknowledgements

This work is supported by the ERC Advanced Grant no. 320651, "HEPGAME"