Evo-MCTS: Gravitational-Wave Algorithm Search and Lessons for AI-Assisted Physics

He Wang

Based on arXiv:2508.03661

hewang@ucas.ac.cn

International Centre for Theoretical Physics – Asia Pacific (ICTP-AP)

University of Chinese Academy of Sciences (UCAS)

25th Oct 2026 | ICFP 2026

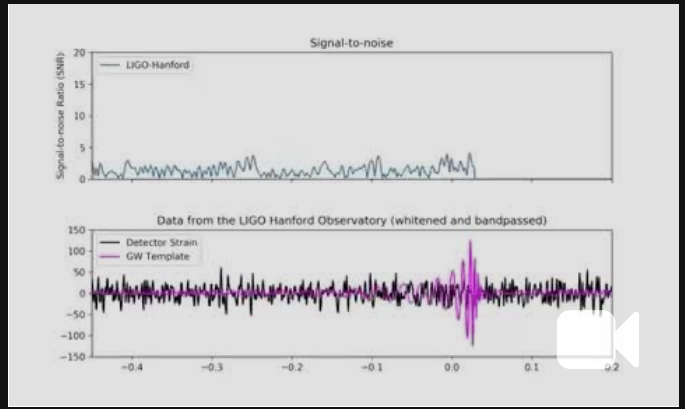
Challenges and Motivations

Matched filtering

- Sensitive to modeled signals, with computational costs that grow with template-bank coverage.

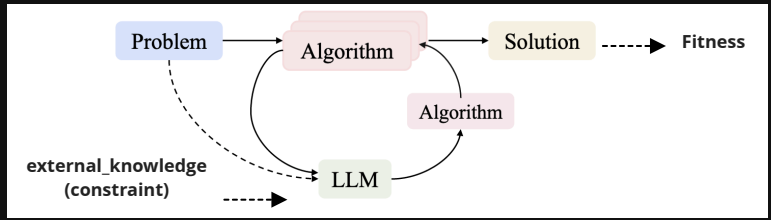
Neural networks

- Fast inference, with additional work needed to assess robustness and interpret learned decisions.



Our question

- Can we search for detection programs that experts can inspect and test?



arXiv:2410.14716 [cs.LG]

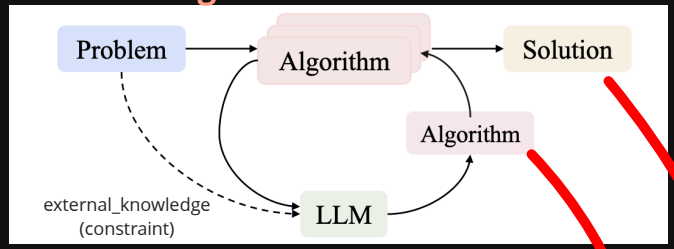
hewang@ucas.ac.cn

Evo-MCTS searches over executable detection pipelines

HW & ZL, arXiv:2508.03661

The LLM does not predict answers — it reshapes how we search for algorithms.

LLM as designer



arXiv:2410.14716 [cs.LG]

```python
import numpy as np
import scipy.signal as signal
def pipeline_v1(strain_h1: np.ndarray, strain_l1: np.ndarray, time
    def data_conditioning(strain_h1: np.ndarray, strain_l1: np.nda
        window_length = 4096
        dt = times[1] - times[0]
        fs = 1.0 / dt

        def whiten_strain(strain):
            strain_zeromean = strain - np.mean(strain)
            freqs, psd = signal.welch(strain_zeromean, fs=fs, nper
                                      window='hann', noverlap=win
            smoothed_psd = np.convolve(psd, np.ones(32) / 32, mode='same')
            smoothed_psd = np.maximum(smoothed_psd, np.finfo(float).tiny)
            white_fft = np.fft.rfft(strain_zeromean) / np.sqrt(np.interp(np.fft.rfftfreq(len(
            return np.fft.irfft(white_fft)

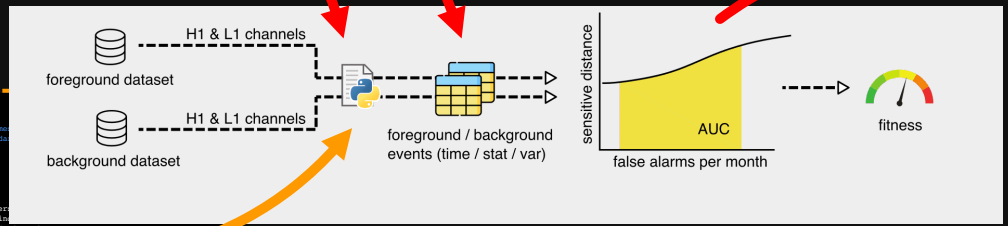
        whitened_h1 = whiten_strain(strain_h1)
        whitened_l1 = whiten_strain(strain_l1)

        return whitened_h1, whitened_l1, times

    def compute_metric_series(h1_data: np.ndarray, l1_data: np.ndarray, time_series: np.ndarray) -> tuple[np.nda
        fs = 1 / (time_series[1] - time_series[0])
        f_h1, t_h1, Sxx_h1 = signal.spectrogram(h1_data, fs=fs, nperseg=256, noverlap=128, mode='magnitude', det
        f_l1, t_l1, Sxx_l1 = signal.spectrogram(l1_data, fs=fs, nperseg=256, noverlap=128, mode='magnitude', det
        tf_metric = np.mean((Sxx_h1**2 + Sxx_l1**2) / 2, axis=0)
        gps_mid_time = time_series[0] + (time_series[-1] - time_series[0]) / 2
        metric_times = gps_mid_time + (t_h1 - t_h1[-1] / 2)

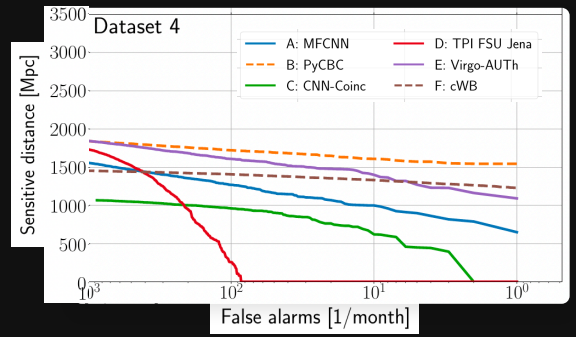
        return tf_metric, metric_times

    def calculate_statistics(tf_metric, t_h1):
        background_level = np.median(tf_metric)
        peaks, _ = signal.find_peaks(tf_metric, height=background_level * 1.0, distance=2, prominence=background
        peak_times = t_h1[peaks]
        peak_heights = tf_metric[peaks]
        peak_deltat = np.full(len(peak_times), 10.0)  # Fixed uncertainty value
        return peak_times, peak_heights, peak_deltat
```



Execute on benchmark data

MLGWSC-1 benchmark



Optimization Target: Maximizing Area Under Curve (AUC) over the target false-alarm-rate range (e.g. 1–1000 false alarms/month), balancing detection sensitivity and false alarm rates across algorithm generations

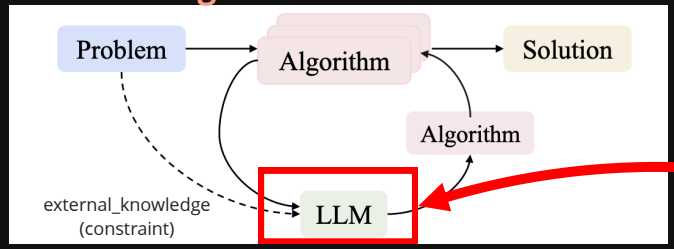
hewang@ucas.ac.cn

Performance feedback guides the next code revision

HW & ZL, arXiv:2508.03661

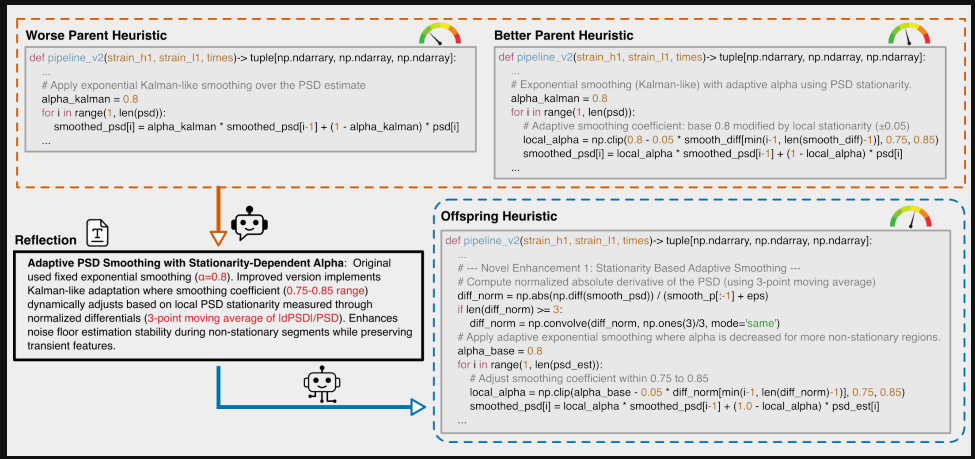
The LLM does not predict answers — it reshapes how we search for algorithms.

LLM as designer



arXiv:2410.14716 [cs.LG]

LLM-Driven Algorithmic Evolution Through Reflective Code Synthesis.



One Prompt Template for MLGWSC1 Algorithm Synthesis

```
You are an expert in gravitational wave signal detection algorithms. Your
task is to design heuristics that can effectively solve optimization
problems.

{prompt_task}

I have analyzed two algorithms and provided a reflection on their
differences.

[Worse code]
{worse_code}

[Better code]
{better_code}

[Reflection]
{reflection}

{external_knowledge}

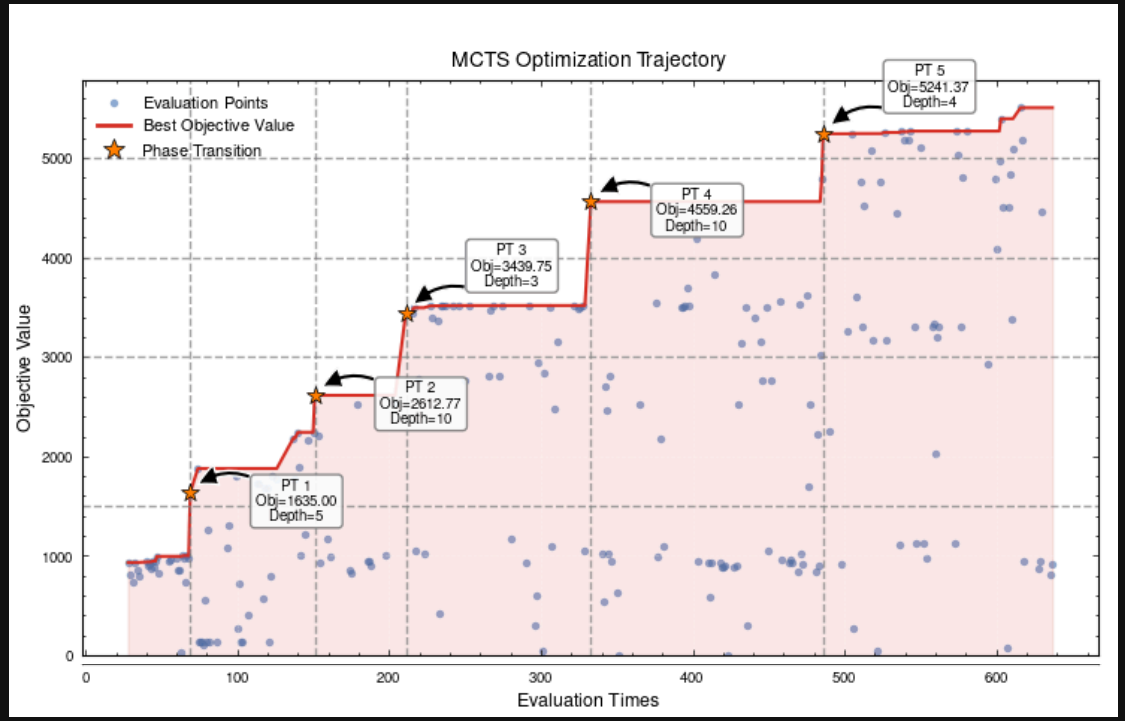
Based on this reflection, please write an improved algorithm according to
the reflection.
First, describe the design idea and main steps of your algorithm in one
sentence. The description must be inside a brace outside the code
implementation. Next, implement it in Python as a function named
'{func_name}'.
This function should accept {input_count} input(s): {joined_inputs}. The
function should return {output_count} output(s): {joined_outputs}.
{inout_inf} {other_inf}

Do not give additional explanations.
```

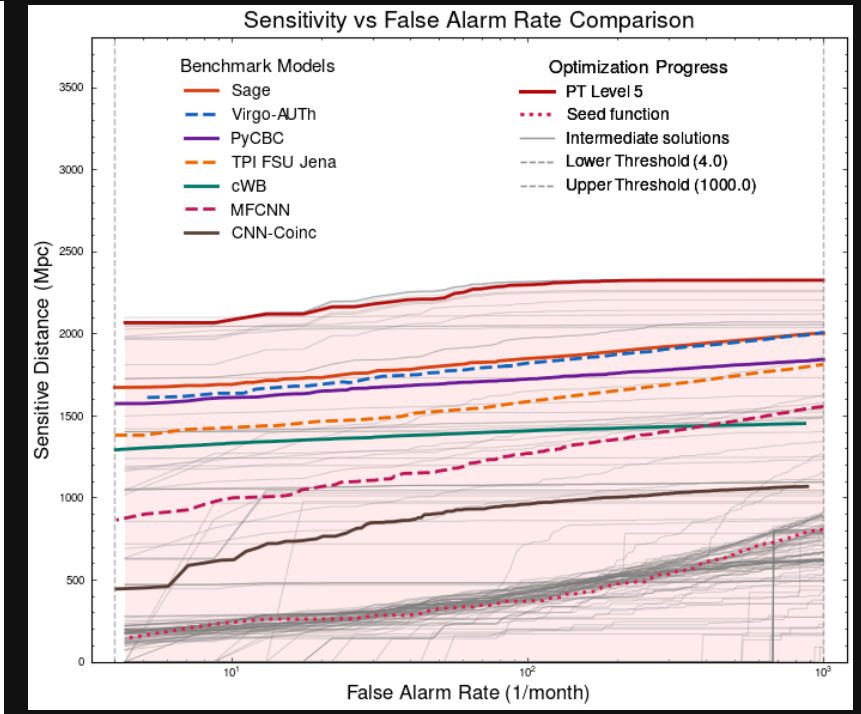
MLGWSC1 Benchmark: Optimization Performance Results

HW & ZL, arXiv:2508.03661

Automated exploration of algorithm parameter space



Benchmarking against state-of-the-art methods



hewang@ucas.ac.cn

MLGWSC1 Benchmark: Optimization Performance Results

HW & ZL, arXiv:2508.03661

Automated exploration of algorithm parameter space

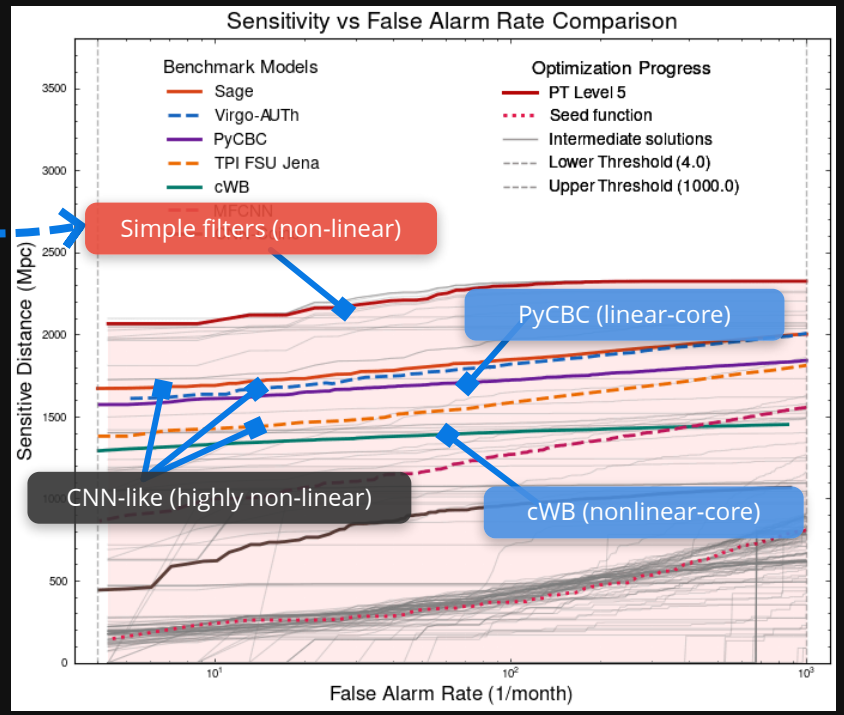
Benchmarking against state-of-the-art methods

Table A1 Performance Comparison Across Benchmark Models and Phase Transition Levels

Model	AUC	Sensitive Distance (Mpc) at FAR			
	(units)	1000	100	10	4.3
Benchmark Models					
Sage	4359.27	1996.1	1846.6	1688.8	1672.4
Virgo-AUTh	4101.48	1990.2	1818.7	1635.0	1609.5
PyGCN	4069.90	1832.3	1721.8	1609.3	1573.4
TPI FSU Jena	3744.99	1	1	426.4	1382.4
CWB	3225.0	1	1	351.8	1303.2
MFCNN	2890.33	1	1	997.2	900.3
CNN-Coinc	1997.02	1	1	620.4	450.8
Phase Transition (PT) Levels					
PT Level 5	5241.37	2323.9	2295.8	2080.9	2065.3
PT Level 4	4559.26	1932.0	1932.0	1932.0	1932.0
PT Level 3	3439.75	1537.2	1460.1	1407.9	1402.3
PT Level 2	2612.77	1107.2	1107.2	1107.2	1107.2
PT Level 1	1635.00	—	769.8	769.8	769.8
Seed function	926.03	786.9	368.2	238.3	158.3

+20.2% benchmark AUC relative to Sage

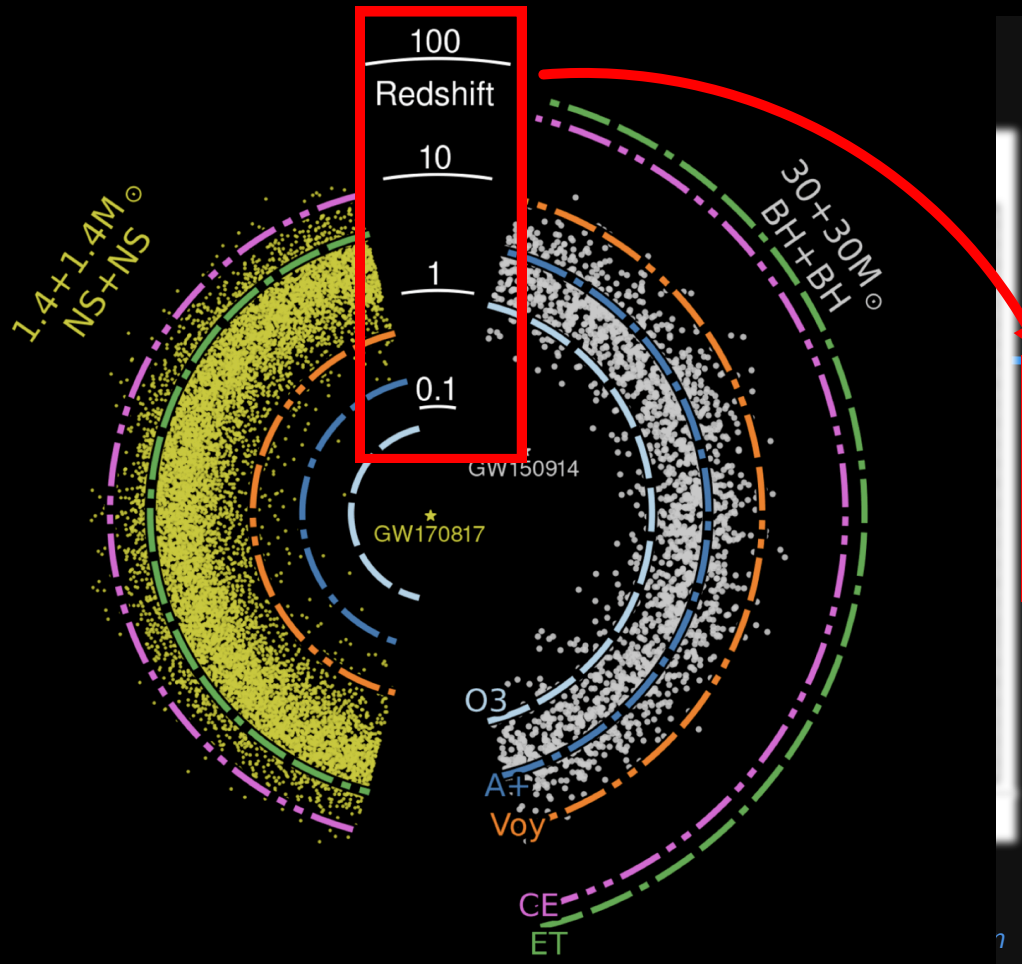
~23% higher sensitive distance at FAR = 4.3/month



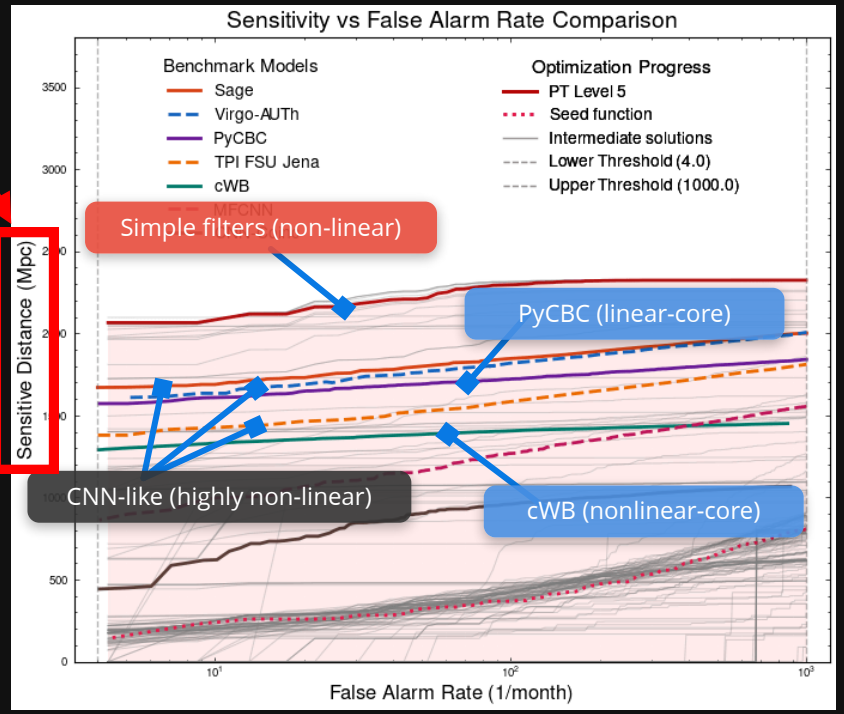
hewang@ucas.ac.cn

MLGWSC1 Benchmark: Optimization Performance Results

HW & ZL, arXiv:2508.03661



Benchmarking against state-of-the-art methods



Understanding the search results

Examining the generated programs and the evidence behind their performance

LLM-Guided Search in Program Space

HW & ZL, arXiv:2508.03661

Example candidate program

```python
import numpy as np
import scipy.signal as signal
from scipy.signal.windows import tukey
from scipy.signal import savgol_filter

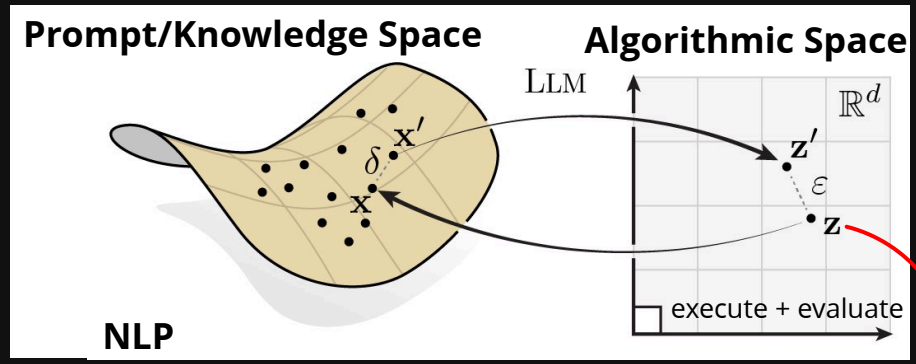
def pipeline_v2(strain_h1: np.ndarray, strain_l1: np.ndarray, times: np.ndarr
    """
    The pipeline function processes gravitational wave data from the H1 and L
    It takes strain_h1 and strain_l1 numpy arrays containing detector data, a
    The function returns a tuple of three numpy arrays: peak_times containing
    peak_heights with significance values of each peak, and peak_deltat show
    """
    eps = np.finfo(float).tiny
    dt = times[1] - times[0]
    fs = 1.0 / dt
    # Base spectrogram parameters
    base_nperseg = 256
    base_noverlap = base_nperseg // 2
    medfilt_kernel = 101        # odd kernel size for robust detrending
    uncertainty_window = 5      # half-window for local timing uncertainty

    # -------------------- Stage 1: Robust Baseline Detrending -------------
    # Remove long-term trends using a median filter for each channel.
    detrended_h1 = strain_h1 - signal.medfilt(strain_h1, kernel_size=medfilt_
    detrended_l1 = strain_l1 - signal.medfilt(strain_l1, kernel_size=medfilt_

    # -------------------- Stage 2: Adaptive Whitening with Enhanced PSD Smoo
    def adaptive_whitening(strain: np.ndarray) -> np.ndarray:
        # Center the signal.
        centered = strain - np.mean(strain)
        n_samples = len(centered)
        # Adaptive window length: between 5 and 30 seconds
        win_length_sec = np.clip(n_samples / fs / 20, 5, 30)
        nperseg_adapt = int(win_length_sec * fs)
```

Lesson 1: Guided brute-force search in program space

- LLMs propose programs rather than directly solving the scientific problem.
- Evolutionary strategies and MCTS guide the search through executable candidates.
- Much of this process can be viewed as **recombination** and "**interpolation**" of existing knowledge.



Intelligent guidance does not eliminate trial-and-error search.

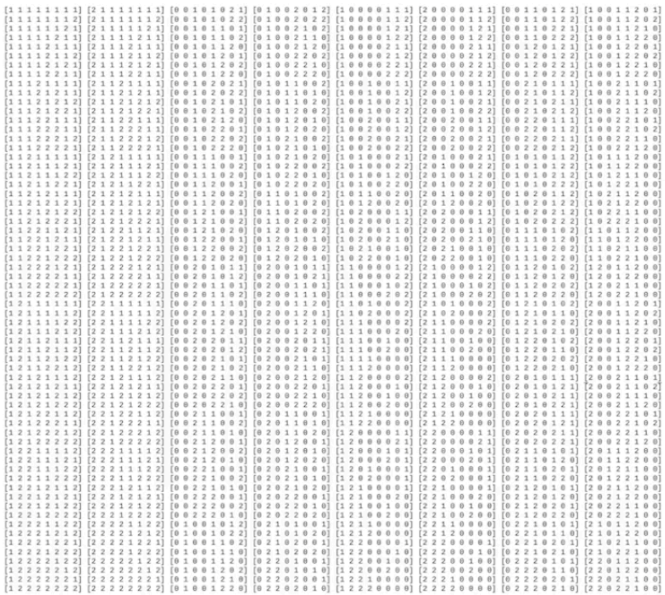
hewang@ucas.ac.cn

From Solutions to Programs: FunSearch

LLM guided search in program space

Solution space vs **function** space

Raw list of nodes



Program that outputs a list of nodes

```python
def get_capset(n: int) -> CapSet:
  """Returns a 512-cap in AG(8, 3)."""
  V = np.array(list(itertools.product(range(3), repeat=n)), dtype=np.int32)
  reflections = lambda v: sum(1 for i in range(1, n // 2) if v[i] == v[-i])

  # First we list 128 weight-8 vectors with >= 2 reflections.
  weight8_points = [v for v in V
                    if np.count_nonzero(v) == 8  # Weight is 8.
                    and reflections(v) >= 2]  # At least 2 reflections.

  # Then we list 256 weight-4 vectors with allowed support and <= 1 reflections.
  allowed_supports = [
      (0, 1, 2, 3), (0, 1, 2, 5), (0, 1, 2, 7), (0, 1, 2, 6), (0, 1, 3, 7),
      (0, 1, 6, 7), (0, 3, 6, 7), (0, 5, 6, 7), (0, 1, 5, 7), (1, 3, 4, 6),
      (1, 4, 5, 6), (0, 2, 3, 6), (2, 3, 4, 7), (2, 4, 5, 7), (0, 2, 6, 7),
      (0, 2, 5, 6), (1, 2, 4, 7), (1, 2, 4, 6), (1, 3, 4, 7), (1, 4, 6, 7),
      (1, 4, 5, 7), (2, 3, 4, 6), (2, 4, 6, 7), (2, 4, 5, 6),
  ]
  weight4_points = [
      v for v in V
      if np.count_nonzero(v) == 4  # Weight is 4.
      and tuple(i for i in range(n) if v[i] != 0) in allowed_supports
      and reflections(v) <= 1]  # At most 1 reflection.

  # Finally we add 128 weight-5 vectors with <= 1 reflections.
  allowed_zeros = [(0, 4, 7), (0, 2, 4), (0, 1, 4), (0, 4, 6),
                   (1, 2, 6), (2, 6, 7), (1, 2, 7), (1, 6, 7)]
  weight5_points = [
      v for v in V
      if np.count_nonzero(v) == 5  # Weight is 4.
      and tuple(i for i in range(n) if v[i] == 0) in allowed_zeros
      and reflections(v) <= 1  # At most 1 reflection.
      and (v[1] * v[7]) % 3 != 1 and (v[2] * v[6]) % 3 != 1]  # Mod conditions.

  return weight8_points + weight4_points + weight5_points
```

Fig. 4: Result of applying FunSearch to the cap set problem.

a

n	3	4	5	6	7	8
Best known	9	20	45	112	236	496
FunSearch	9	20	45	112	236	512

nature | Google DeepMind

Explore content | About the journal | Publish with us

nature › articles › article

Article | Open access | Published: 14 December 2023

Mathematical discoveries from program search with large language models

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli & Alhussein Fawzi

Nature **625**, 468–475 (2024) | Cite this article

247k Accesses | **71** Citations | **1027** Altmetric | Metrics

YouTube (Source)

Tracing the Evolution of a Detection Program

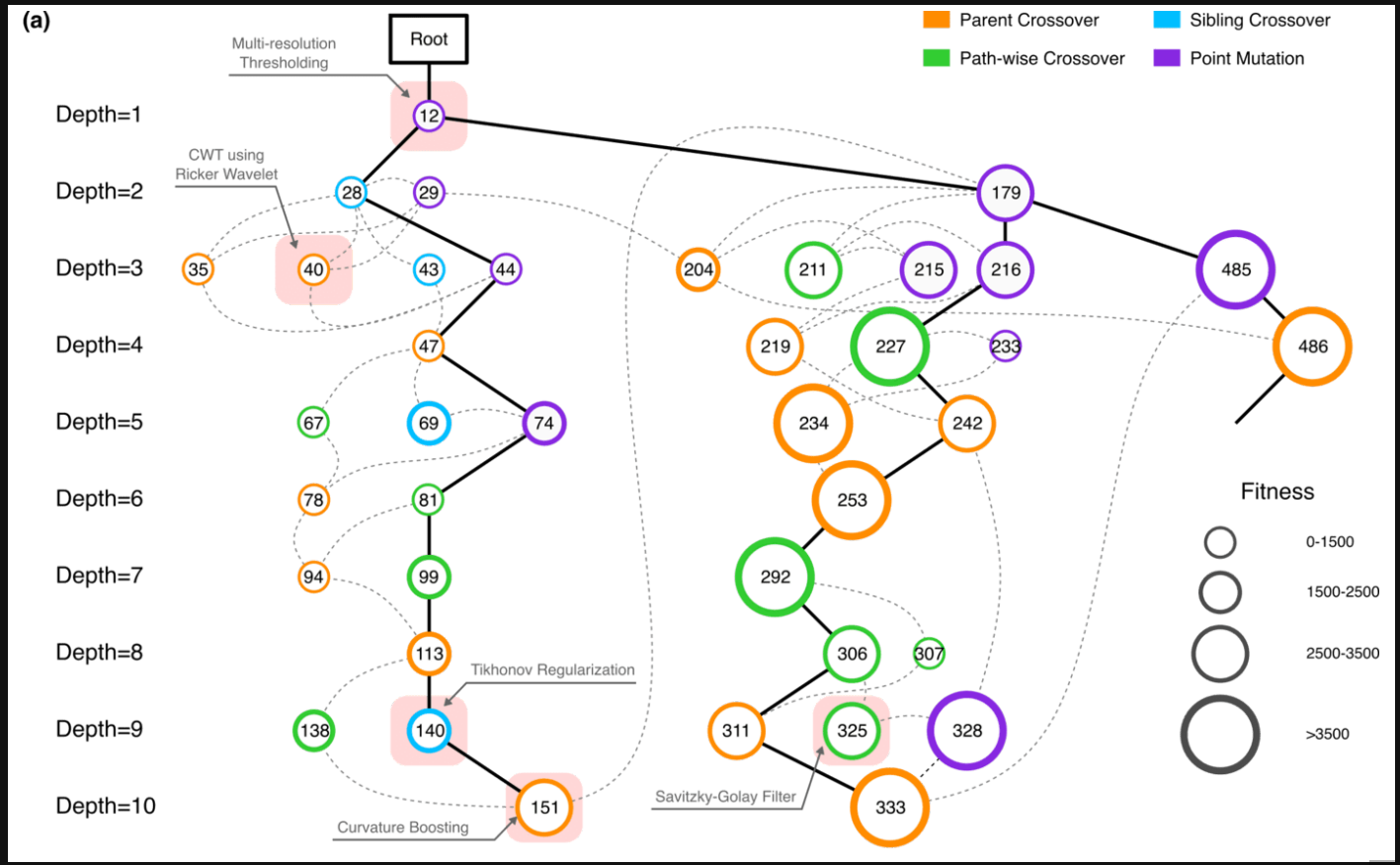
HW & ZL, arXiv:2508.03661

Evolutionary search guided by MCTS

- Each node represents an executable candidate program.
- Colors indicate different evolutionary operations.
- Node size reflects benchmark fitness.

- **The search history reveals how algorithms evolve through successive revisions.**

The search history reveals how algorithms evolve through successive revisions.



hewang@ucas.ac.cn

Evo-MCTS: An Early Form of AI Research Agent

HW & ZL, arXiv:2508.03661

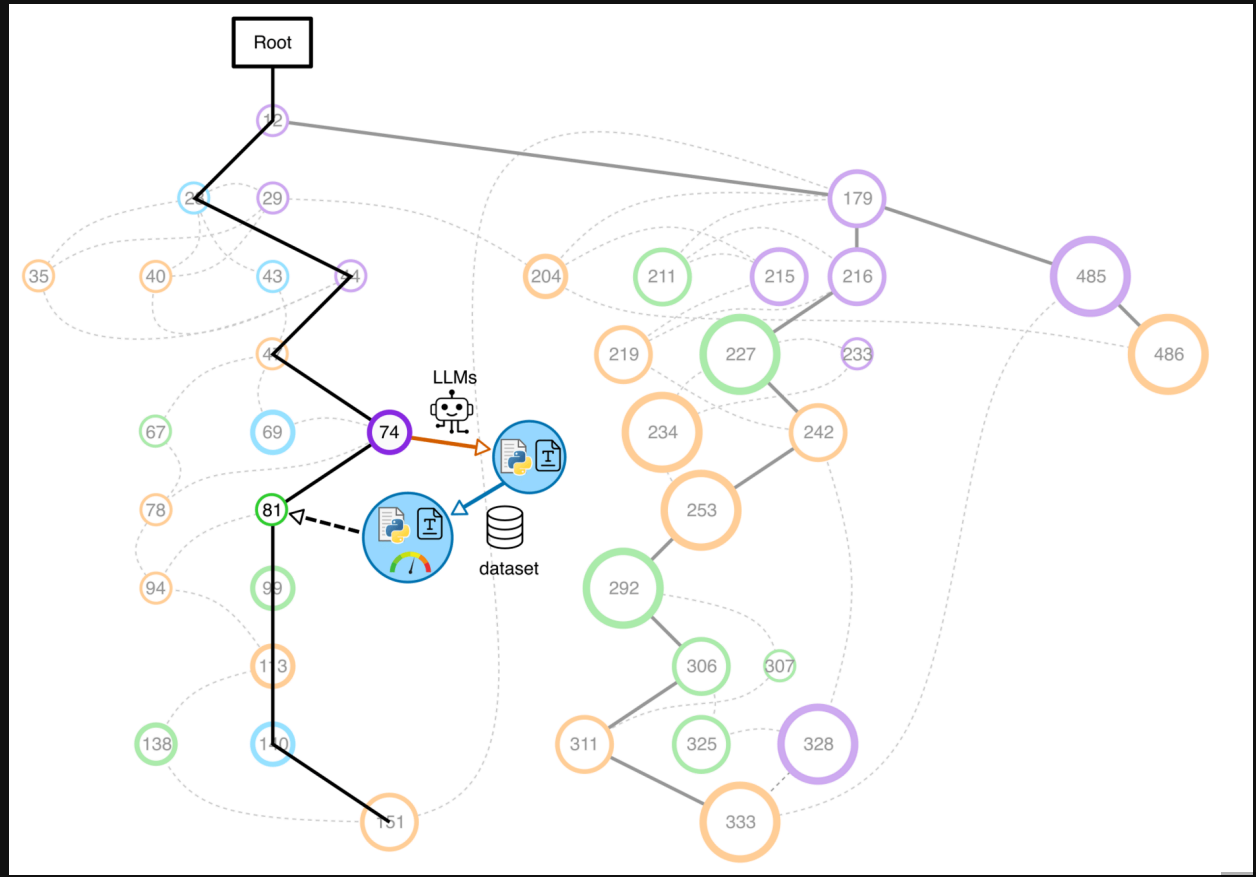
Developed in 2024, Evo-MCTS combined evolutionary algorithms, Monte Carlo Tree Search, and LLM-driven code generation.

Lesson 2: In retrospect, it already embodied the core architecture of today's **AI research agents.**

AI Agent = LLMs + Harness

Harness = Search + Execution + Evaluation + Feedback

Toward self-improving research agents: Can the search process itself become an optimization target?



hewang@ucas.ac.cn

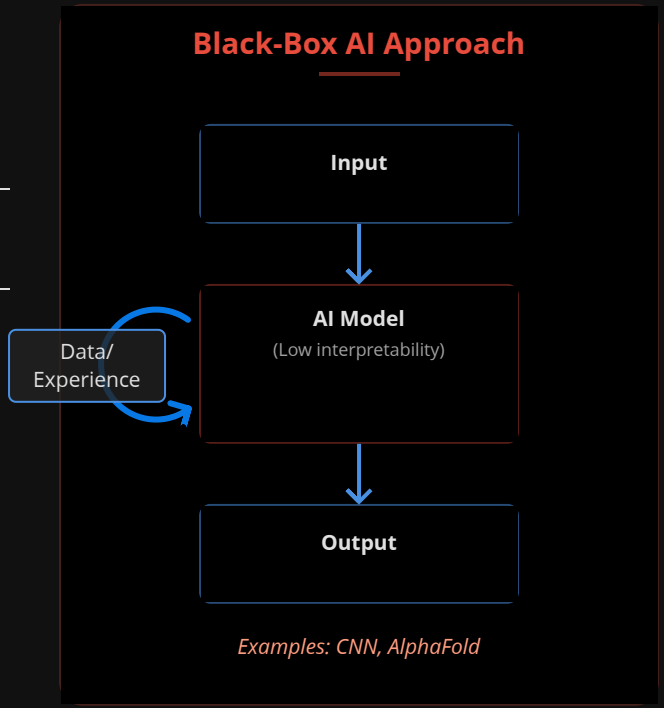
From Predictive Models to Algorithm Discovery

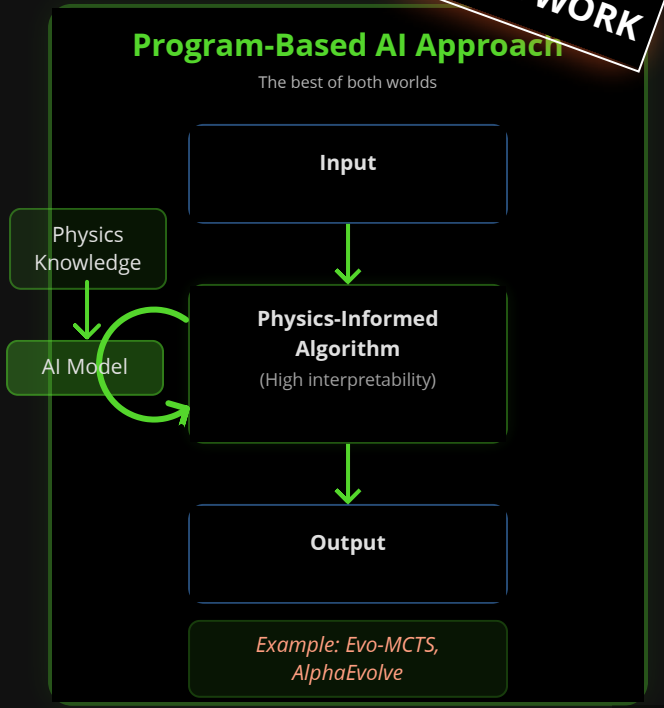
HW & ZL, arXiv:2508.03661

Lesson 3: AI can optimize not only model parameters, but also the structure of scientific algorithms.

Learning a predictor	Searching for a program
Optimize model parameters	Optimize program structure and parameters
Produce a trained model	Produce executable analysis code
Examine model behavior	Inspect code and test individual components

Explicit programs enable inspection, modification, and targeted validation.





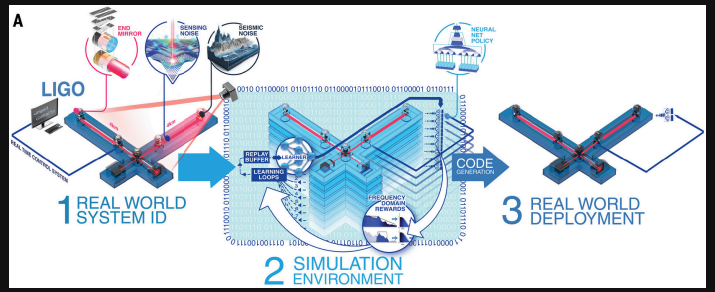
hewang@ucas.ac.cn

Searching for Scientific Discoveries

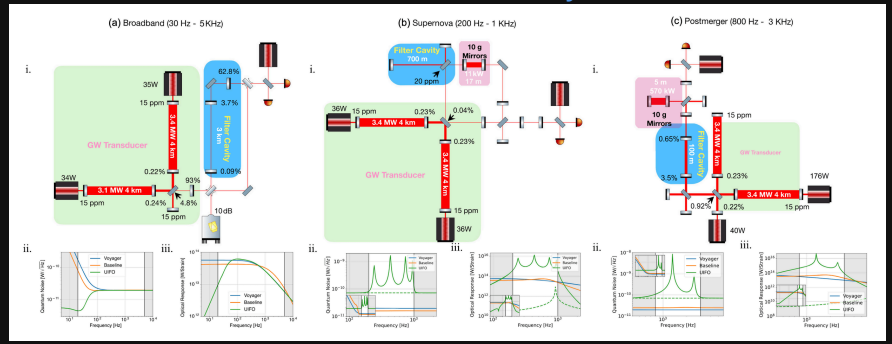
Scientific discovery involves many processes that can be formulated as search and optimization problems.

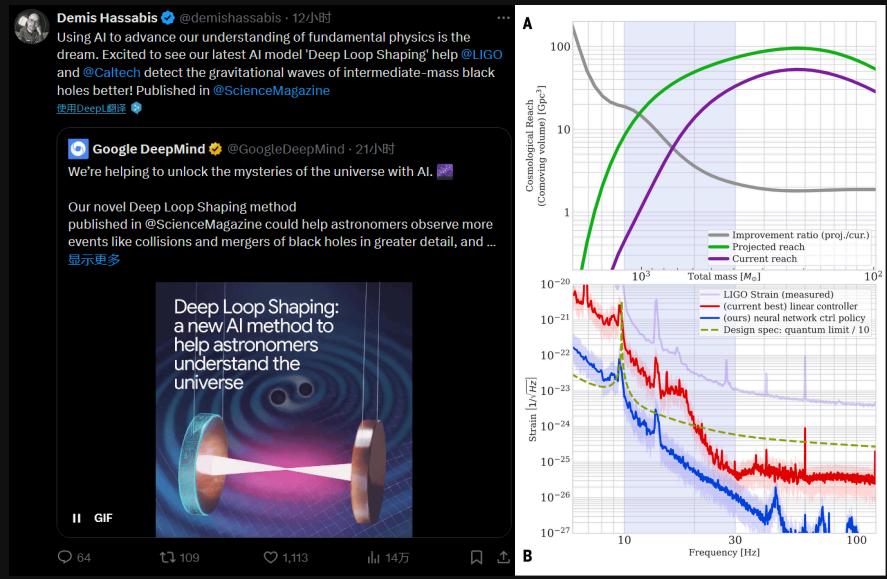
- **Case 1 — AI-driven experiment design**
  - Experiment design as optimization
  - The search object is no longer a parameter vector — it can be the experiment itself.
- **Case 2 — Deep Loop Shaping for LIGO**
  - simulation-trained controller → real LIGO deployment
  - The discovered object is a control policy, not a prediction.

Buchli et al., *Science* 389, 1012–1015 (2025).



Krenn, Drori & Adhikari, *Phys. Rev. X* 15, 021012 (2025).





Scientific discovery itself may still offer vast, largely unexplored opportunities for optimization.

Takeaways: From Algorithm Search to AI-Assisted Science

Takeaways

1. **Guided Search, Not Magic**

   LLM-driven discovery remains largely a guided search over programs, building on existing knowledge.

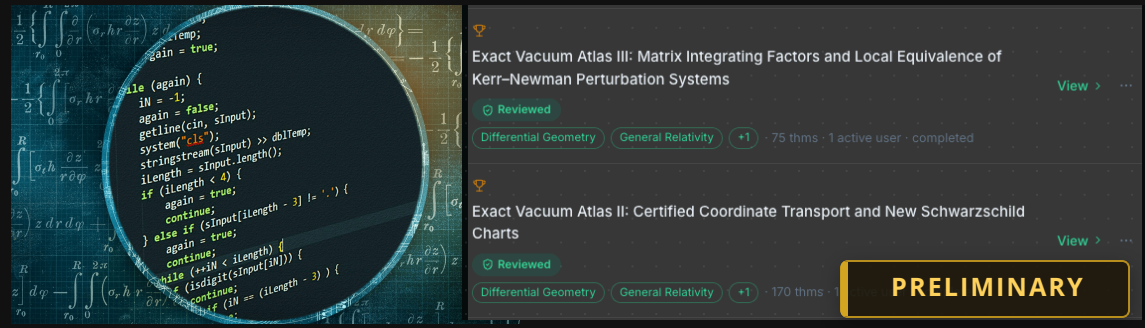
2. **From Evo-MCTS to Research Agents**

   Evolutionary search, MCTS, LLMs, and evaluation harnesses already form the basis of an autonomous research loop.

3. **Beyond Algorithm Discovery**

   Scientific research may contain far more optimizable structures and processes than we currently exploit.

Completed: Lean verification of matrix integrating-factor identities and local equivalence for Kerr–Newman perturbation systems. **Ref: *Hintz* (2026)**



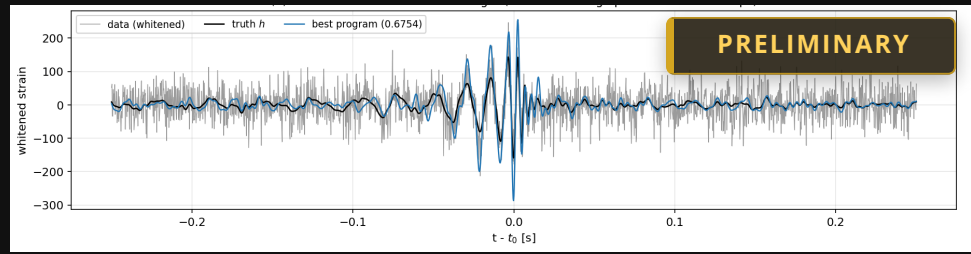
Current Research

- **Formal Verification in Theoretical Physics**

  Lean 4 formalization of black-hole solutions and perturbation identities.

- **Automated Gravitational-Wave Algorithm Discovery**

  Searching for improved event-specific denoising and signal-processing strategies.



Our aim is to connect automated search with independently checkable scientific claims.

hewang@ucas.ac.cn

unesco
Centre
Under the auspices
of UNESCO

ICTP-AP
INTERNATIONAL CENTRE FOR
THEORETICAL PHYSICS ASIA-PACIFIC

University of Chinese Academy of Sciences

```
for _ in range(num_of_audiences):
    print('Thank you for your attention! 🙏')
```

hewang@ucas.ac.cn

CALL FOR PAPERS



Collection details

the-innovation.org/the-innovation-astronomy/Collections/gwagh

Submit your manuscript

Gravitational-Wave Astronomy

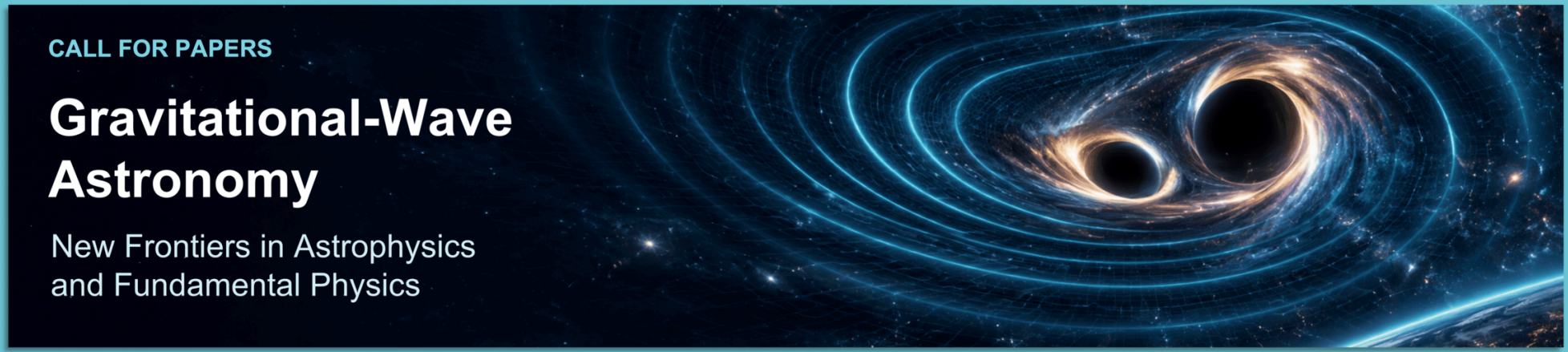
New Frontiers in Astrophysics and Fundamental Physics

Connecting source and detector science with astrophysical discovery and fundamental physics.

COLLECTION SCOPE

01 Sources and astrophysical populations
02 Relativistic dynamics, waveforms and simulations
03 Data analysis, statistical inference and AI
04 Tests of gravity and fundamental physics
05 Cosmology, stochastic backgrounds and the early Universe
06 Multiband and multimessenger astronomy
07 Detector science, technologies and future observatories

Original research, reviews, perspectives and commentaries welcome



CALL FOR PAPERS

Gravitational-Wave Astronomy

New Frontiers in Astrophysics and Fundamental Physics

COLLECTION SCOPE

01 Sources and astrophysical populations

02 Relativistic dynamics, waveforms and numerical simulations

03 Data analysis, statistical inference and AI

04 Tests of gravity and fundamental physics

05 Cosmology, stochastic backgrounds and the early Universe

06 Multiband and multimessenger astronomy

07 Detector science, technologies and future observatories

Original research, reviews, perspectives and commentaries welcome

Collection details

Submit manuscript