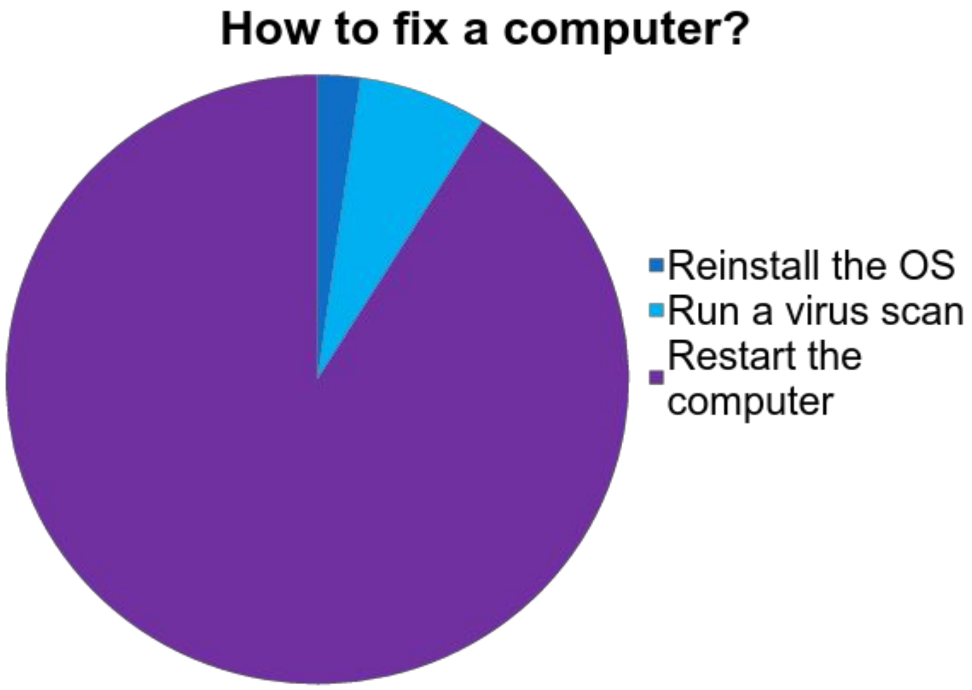
THEORETICAL RESULTS ON BET-AND-RUN AS AN INITIALISATION STRATEGY

Andrei Lissovoi (UoS), Dirk Sudholt, (UoS)
Markus Wagner (UoA), and Christine Zarges (AU)

HOUSTON, WE HAVE A PROBLEM…

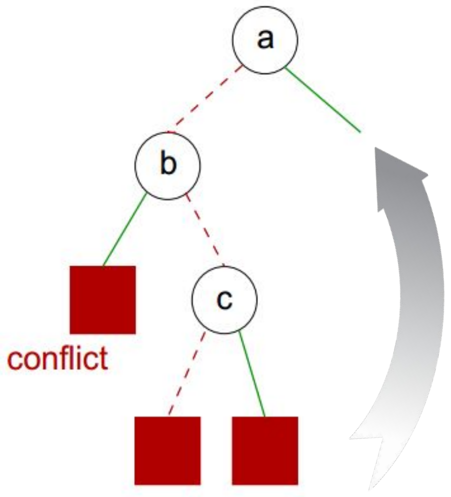
Restarts to the rescue!

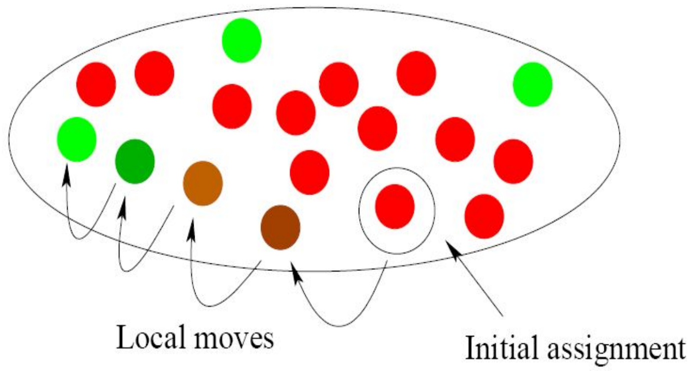


BACKGROUND

Restarted Search

- Become integral part of combinatorial search
- **Complete methods:** avoid heavy-tailed distribution (Gomes et al. JAR'00)
- **Incomplete methods:** diversification technique



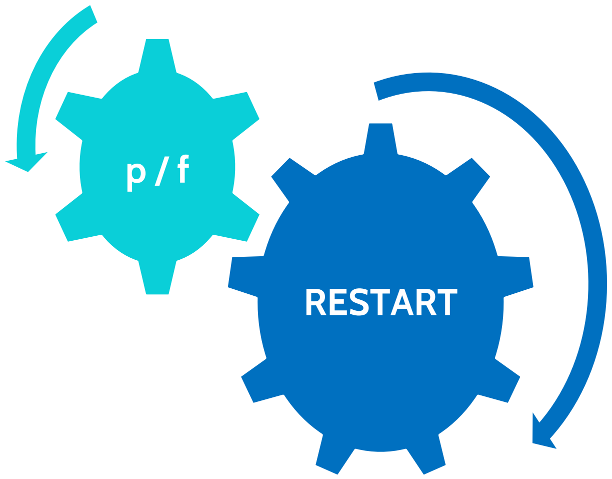


RESTARTS: BACKGROUND

BACKGROUND

Restart Strategies

- ➢ **Complexity** of designing appropriate restart strategy
- ➢ **Two common** approaches:
  1. Use restarts with a certain **probability**
  2. Employ **fixed** schedule of restarts



BACKGROUND

Restart Strategies – Feasibility

- ➢ Theoretical work on fixed-schedule restart strategies (Luby et al.'93)
- ➢ Practical studies with SAT and CP solvers
- ➢ Geometrically growing restarts limits (Wu et al. CP'07)
- ➢ (Audemard et al. CP'12) argued fixed schedules are sub-optimal for SAT

Restart Strategies – Optimization

- ➢ Classical optimization algorithms are often deterministic
  As such, does not really benefit from restarts
- ➢ Modern optimization algorithms have randomized components
  Memory constraints & parallel computation introduce new characteristics
- ➢ (Ruiz et al.'16) different mathematical programming formulations to provide different starting points for the solver

LIMITED RUNTIME BUDGET

Restart Strategies

➢ Assume we are given a **time budget *t*** to run an algorithm

LIMITED RUNTIME BUDGET

Restart Strategies

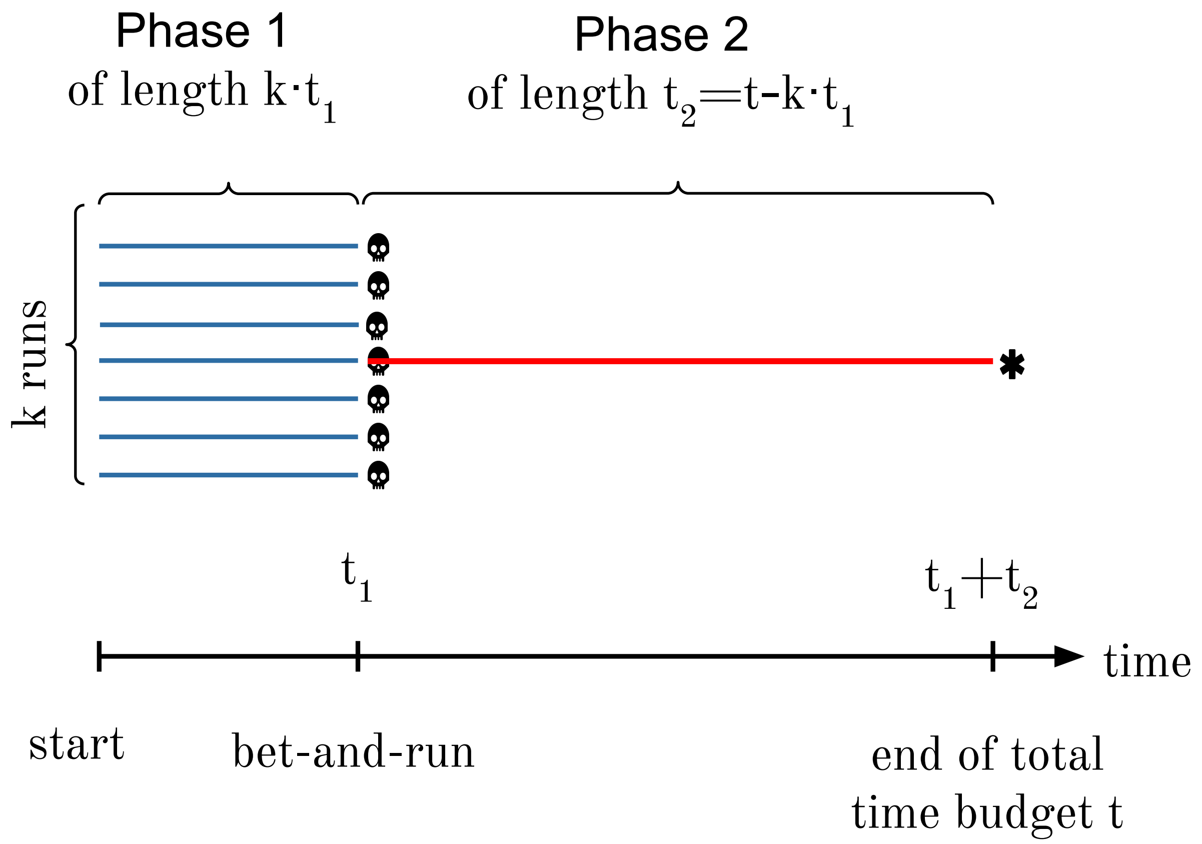
- Assume we are given a time budget *t* to run an algorithm
- Two natural options:
  1. **Single–run strategy:** use all of the time *t* for a single run of the algorithm
  2. **Multi–run strategy:** make *k* runs each with runtime *t/k*

LIMITED RUNTIME BUDGET

Restart Strategies

- Assume we are given a time budget *t* to run an algorithm
- Two natural options:
  1. **Single–run strategy:** use all of the time *t* for a single run of the algorithm
  2. **Multi–run strategy:** make *k* runs each with runtime *t/k*
- (Fischetti et al.'14) generalizes this strategy into **Bet–And–Run** for MIPs

LIMITED RUNTIME BUDGET
BET-AND-RUN BY FISCHETTI AND MONACI (2014)



Another way to interpret this:
degenerated island model, without migration, and the greedy removal of islands

BET–AND–RUN: recent related work

Sampling Phase + Long Run

- ➢ (Fischetti et al. OR'14) introduced diversity in starting conditions of MIP Experimentally good results with $k = 5$
- ➢ (de Perthuis de Laillevault et al. GECCO'15) analysed 1+1-EA on OneMax, t_1=*1step*. A small additive runtime gain, hardly noticeable in practice.
- ➢ (Friedrich, Kötzing, Wagner AAAI'17) studied TSP and MVC Experimentally good results with $\text{Restarts}_{1\%}^{40}$
- ➢ (Kadioglu, Sellmann, Wagner LION'17) learned reactive restart strategies that considers runtime features.
- ➢ (**Lissovoi, Sudholt, Wagner, Zarges GECCO'17**) theoretical results for a family of pseudo-boolean functions. **Summary: non-trivial k and t_1 are necessary** to find the global optimum efficiently.

THEORY

OUTLINE

We analyse the Bet-And-Run strategy:

- with Randomised Local Search (and in some cases a (1+1) EA)
- on a simple artificial benchmark function.

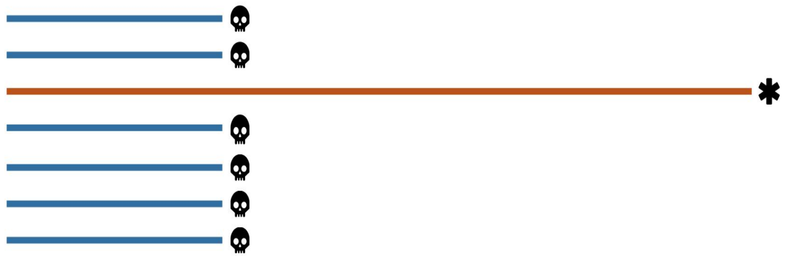
Aiming to answer:

- How does the algorithm behave with given k, t_1, t_2?
- Expected time to find the optimum?
- Expected fitness after $t = k \cdot t_1 + t_2$ iterations?
- How to choose t_1 and k?

BET-AND-RUN and RANDOMISED LOCAL SEARCH

Given a budget of $t = k \cdot t_1 + t_2$ fitness evaluations:

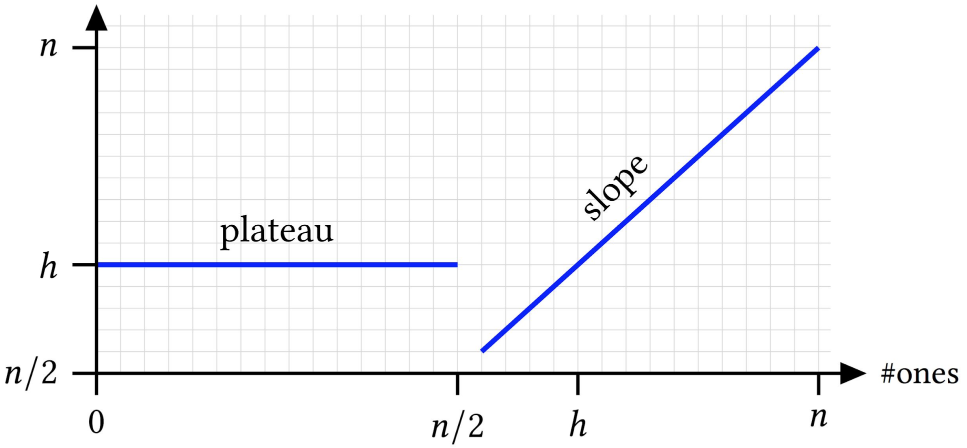
```
1. Run k instances of RLS independently for t_1 steps:
    a. Initialise a solution x uniformly at random.
    b. for i = 2 to t_1 do
        i. Let y be a mutation of x, flipping one bit chosen uniformly at random.
        ii. If f(y) ≥ f(x), replace x with y.
2. Choose run with highest fitness f(x).
3. Continue only this run for another t_2 steps.
```



PLATEAU / SLOPE FUNCTION FAMILY

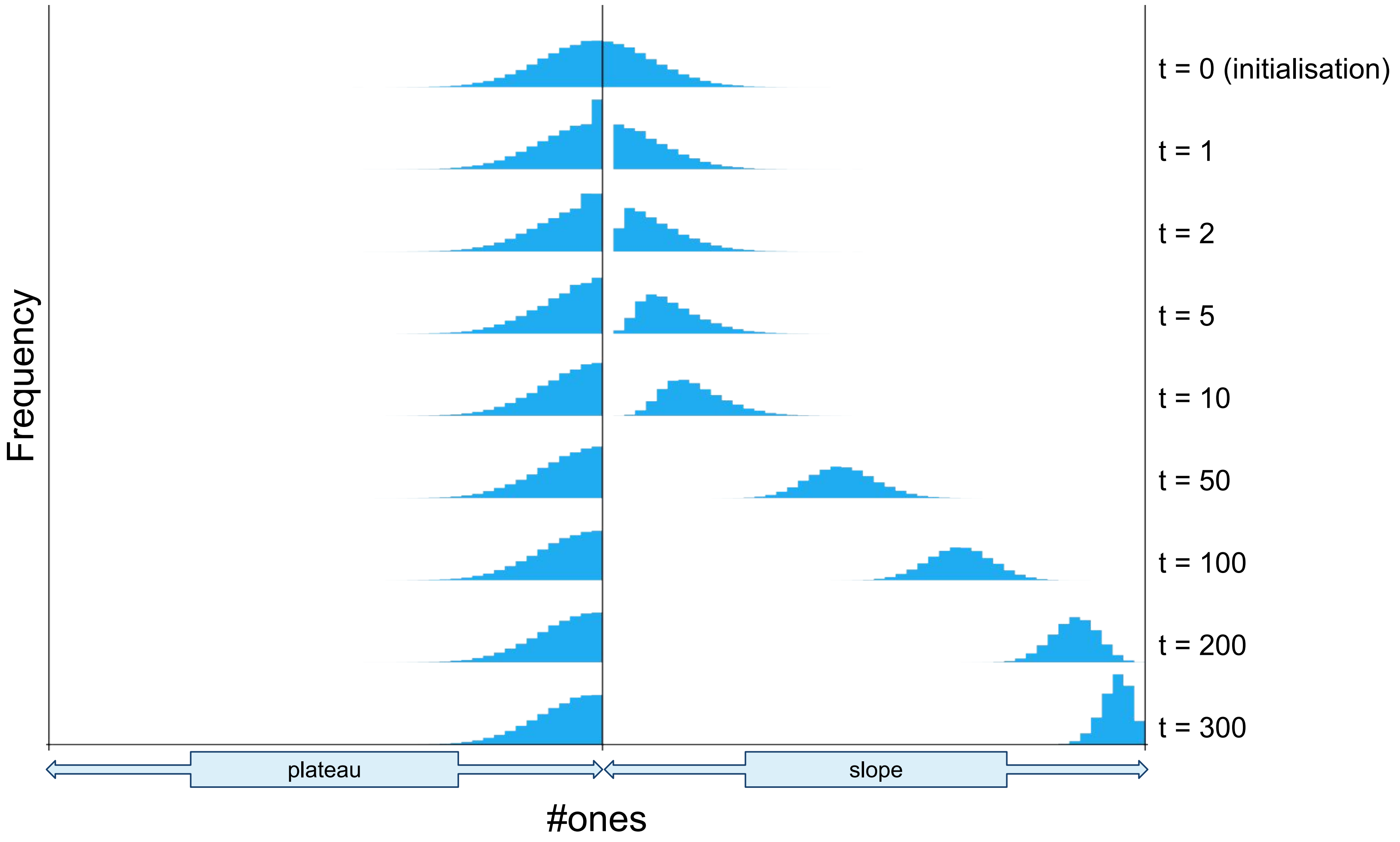
- Individuals are strings of n bits.
- Number of 1-bits affects fitness:
  - Plateau of fitness h when $|\mathbf{x}|_1 \leq n/2$
  - Slope when $|\mathbf{x}|_1 > n/2$
- Family characterised by $h > n/2$

- The plateau is easy to find...
  - ... and hard to escape from.
- The slope is initially worse...
  - ... but leads to the optimum.



$$f_h(x) = \begin{cases} |x|_1 & \text{if } |x|_1 > n/2 \\ h & \text{otherwise} \end{cases}$$

A SINGLE RUN OF RLS



INITIAL PHASE MUST BE LONG ENOUGH

When t_1 is large enough, an on-slope run will climb above the plateau.

Consider f_h with $h > n/2 + n^{0.5} \log n$. For any constant $\varepsilon > 0$,

- If $t_1 \geq (1+\varepsilon)\, n \ln(n/(2n - 2h))$, (and $k \geq c \log n$ for a constant $c > 0$,)
  With probability at least $1 - (3/4)^k - O(1/n)$, the optimum is found after $O(kn \log n)$ fitness evaluations.
- If $t_1 \leq (1-\varepsilon)\, n \ln(n/(2n - 2h))$, (and $k \leq \text{poly}(n)$,)
  With probability at least $1 - 2^{-k} - e^{-\Omega(\sqrt{n})}$, the optimum is **never** found.

The proof uses Fitness Levels with Tail Bounds (Witt '14).

FIXED BUDGET ANALYSIS OF A SINGLE RLS RUN

Where do we expect to be after t iterations?

- If initialised on the plateau, still on the plateau.
- If initialised "safely" on the slope, some distance up the slope.
  - Fixed budget analysis of RLS on OneMax (Jansen/Zarges '14) applies in this case.
- If initialised on the first point of the slope, split *almost* equally.
  - It is slightly easier to get to the plateau.

Combined, the expected fitness after t iterations of a single RLS run is:

- $E(f_h(x_t)) \geq n/2 + h/2 - (n/4 - 1) \cdot (1 - 1/n)^t$
- $E(f_h(x_t)) \leq n/2 + h/2 - (n/4 - 0.5\, n^{0.5} \log n) \cdot (1 - 1/n)^t + \Omega(n^{0.5})$

FIXED BUDGET FOR BET-AND-RUN

When **k** and **t_1** are sufficiently large, at least one run reaches $f_h(x_{t_1}) > h$ with high probability. We bound the expected fitness of the bet-and-run strategy using the fitness achieved by a slope run after t_1+t_2 iterations.

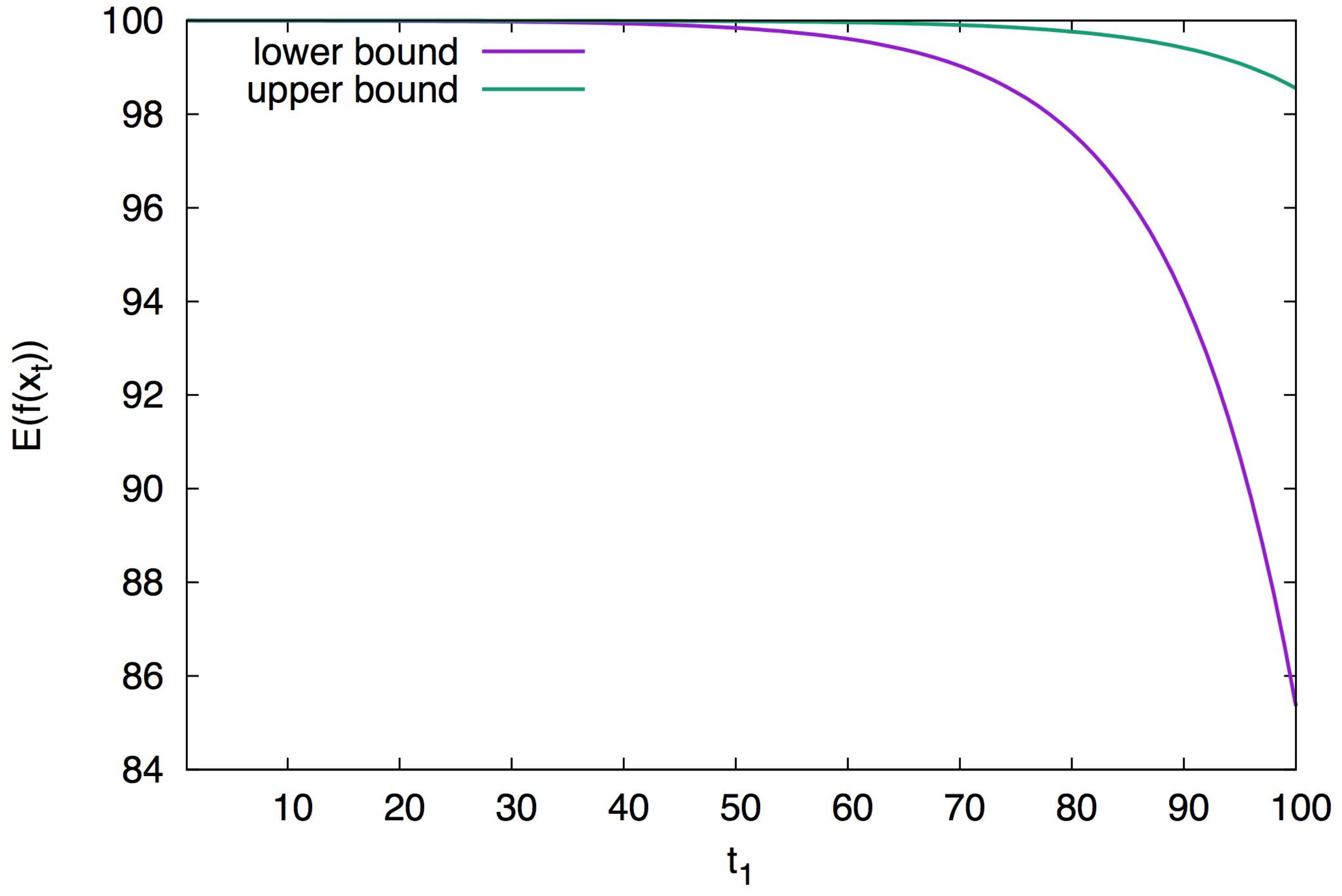
The expected fitness of RLS with a bet-and-run strategy, using $c \log n \leq k \leq \text{poly}(n)$ and $t_1 \geq (1+\varepsilon)n \ln(n/(2n-2h))$, after $t = k \cdot t_1 + t_2$ steps is:

- $E(f(x)) \geq n - (n/2 - d\, n^{0.5}) \cdot (1 - 1/n)^{t - (k-1)t_1} - (3/4)^k\, n$
- $E(f(x)) \leq (1+\square)\, (n - (n/2 - n^{0.5} \log n) \cdot (1 - 1/n)^{t - (k-1)t_1}) + o(1)$

for all $t \geq 0$, and d, $\square$, $\varepsilon > 0$ constant.

Consequence: should not set **t_1** or **k** excessively large.

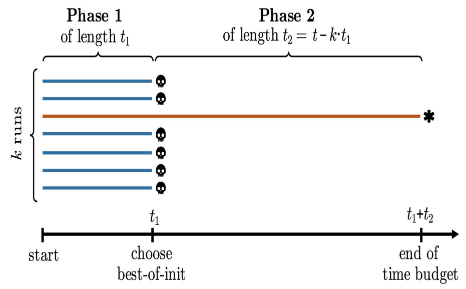
EXCESSIVE T_1 IS DETRIMENTAL

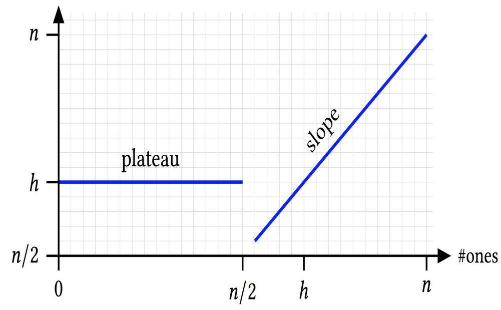


SUMMARY

SUMMARY

- Mathematically proven: bet-and-run can be an effective countermeasure when facing problems with deceptive regions.
- Complementary experiments are in the paper.

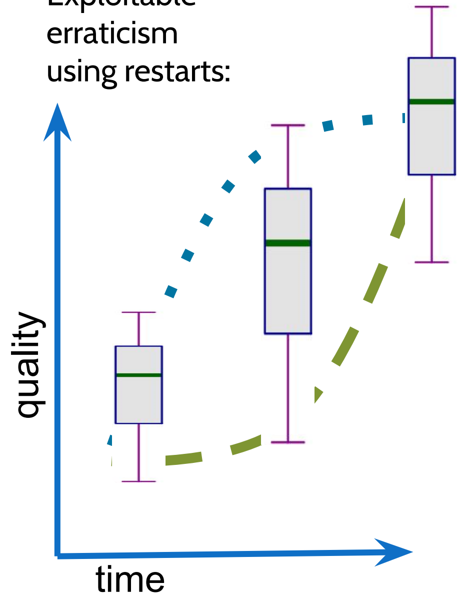




Future work

- Multi-modal functions
- Characterise progress variance of runs in Phase 1 so that this can be exploited in theory and practise.

Exploitable erraticism using restarts:



Special Issue on

„Algorithm Selection and Configuration in Evolutionary Computation“

— Submission Deadline: November 30, 2017 —

JOURNAL:

- Evolutionary Computation Journal
- MIT Press (http://ecj.napier.ac.uk)

POSSIBLE TOPICS (not limited to those):

- automated algorithm selection
- specific machine learning concepts
- configuration methods
- performance analysis
- features and diversity problem instances
- benchmarking concepts
- exploratory landscape analysis

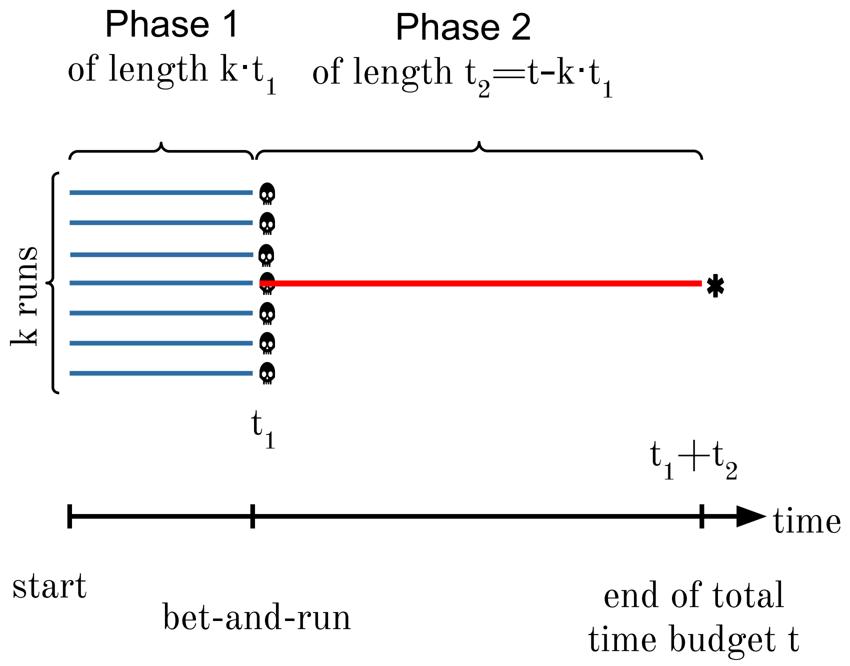
Frank Neumann
University of Adelaide
(Associate Editor)

Holger H. Hoos
Universiteit Leiden
(Guest Editor)

Heike Trautmann
University of Münster
(Guest Editor)

Authors should submit their manuscripts to the Evolutionary Computation Editorial Manager at http://ecj.napier.ac.uk. When submitting a paper, please send at the same time an email to Frank Neumann (frank.neumann@adelaide.edu.au) and a copy to ecj@napier.ac.uk mentioning the special issue, the paper id, title, and author list to inform us about the submission.

LIMITED RUNTIME BUDGET
BET-AND-RUN BY FISCHETTI AND MONACI (2014)



Notes
Single-run:
  k=1
Multi-run with restarts from scratch:
  $t_1=t/k$ and $t_2=0$

Another way to interpret this:
degenerated island model, without migration, and the greedy removal of islands