HNCO
Visualization of empirical autocorrelation functions of various functions defined on bit vectors

May 18, 2020

Abstract

This document proposes to visualize empirical autocorrelation functions of various functions defined on bit vectors (hypercube) of size $n = 100$. If f is a fitness function, a random walk $(X_t)_{t\geq 1}$ on the hypercube gives rise to a time series $(f(X_t))$ which is analyzed through its empirical autocorrelation function.

Contents

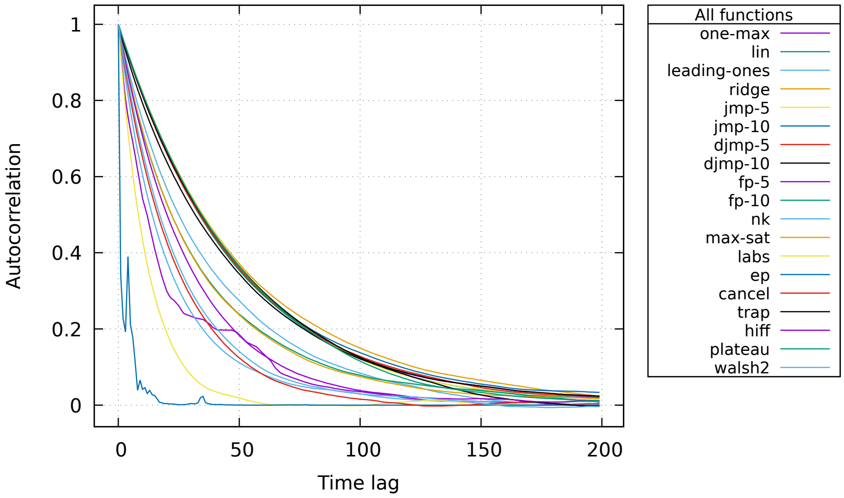
1 Introduction

The underlying process is a random walk $(X_t)_{t\geq 1}$ on the hypercube initialized uniformly. If f is the fitness function then its autocorrelation function ρ is defined by

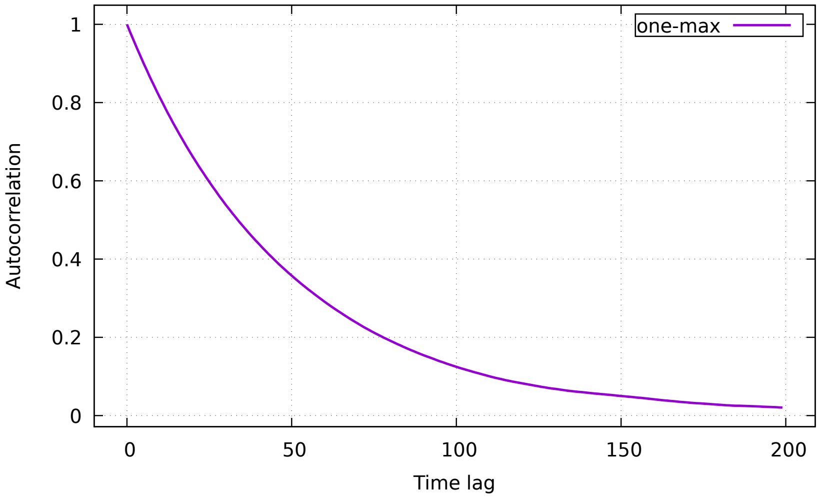
$$\rho(\tau) = \frac{1}{(n-\tau)\sigma^2} \sum_{t=1}^{T-\tau} (f(X_t) - \mu)(f(X_{t+\tau}) - \mu) \tag{1}$$

where μ and σ are the mean and standard deviation respectively of the process $(f(X_t))$, T is the length of the Markov chain and the lag τ is such that $0 \leq \tau < T$. The empirical autocorrelation function is estimated and computed in a naive way. It should be noted that the estimated function does not necessarily have properties such as positivity, monotonicity, or convexity. Normalized autocorrelation functions $\rho(\tau)/\rho(0)$ are represented in the following sections.

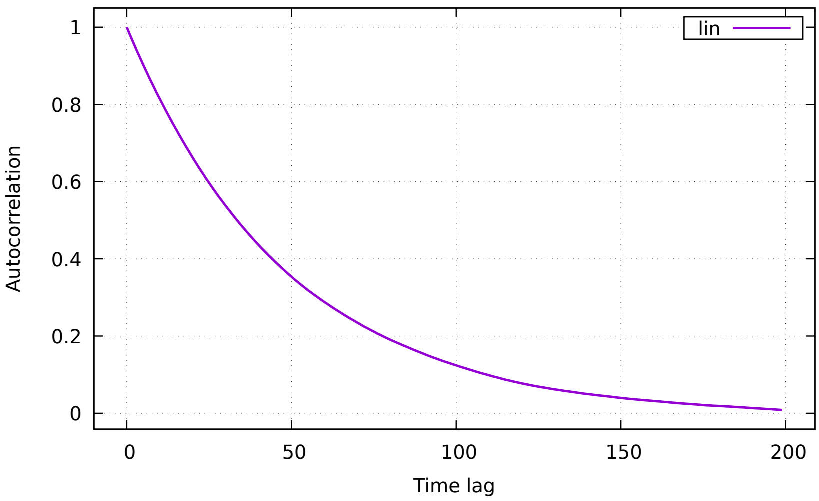
2 All functions



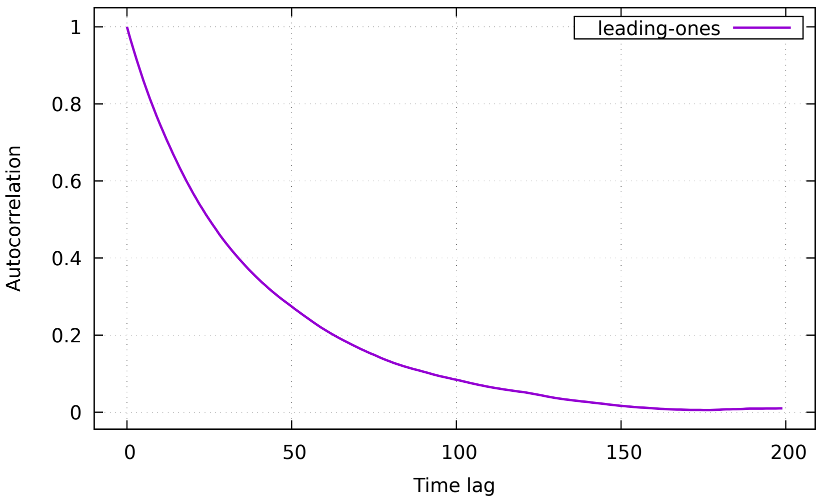
3 one-max



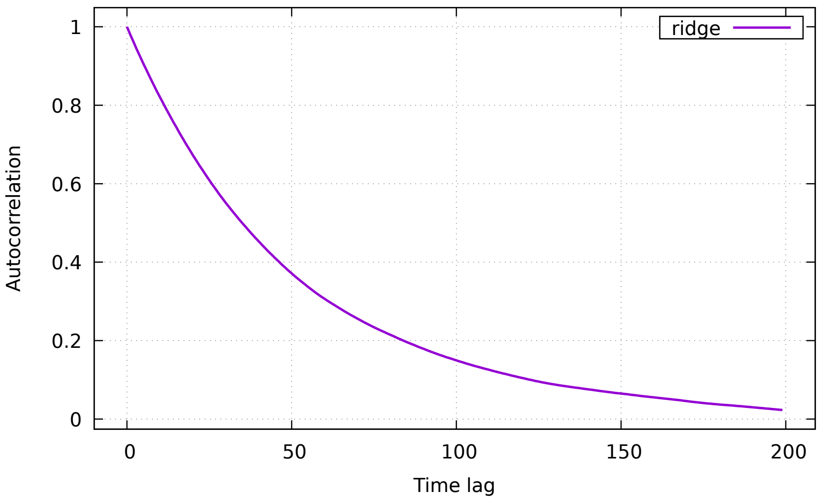
4 lin



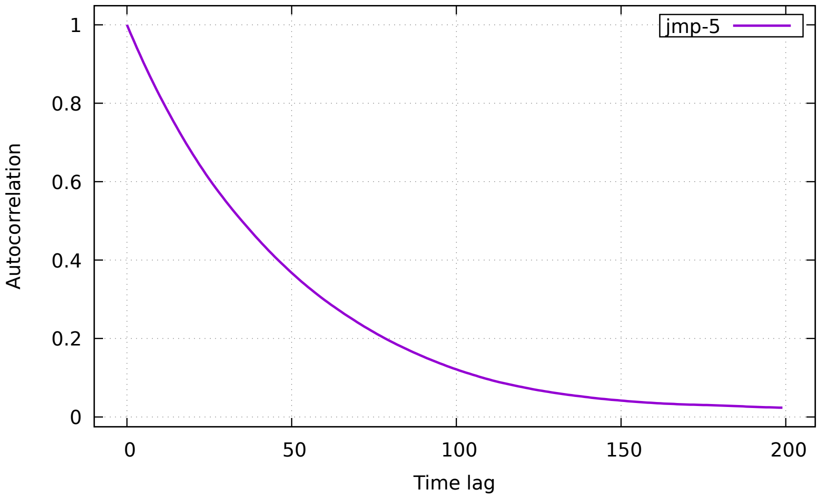
5 leading-ones



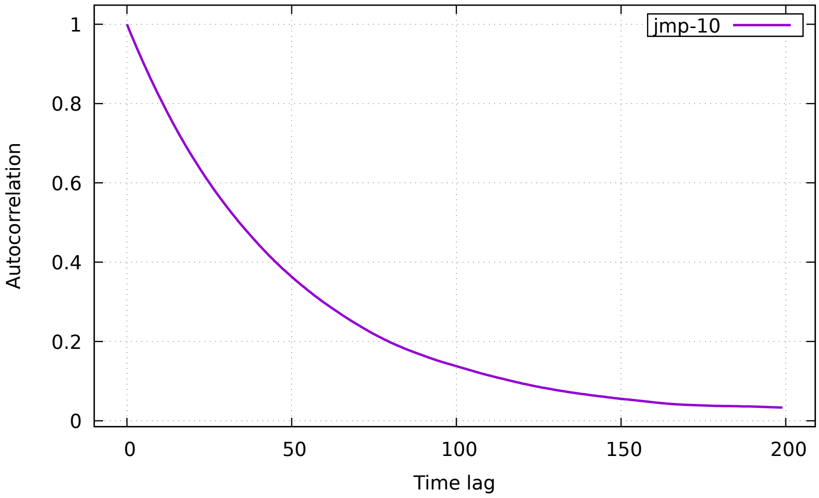
6 ridge



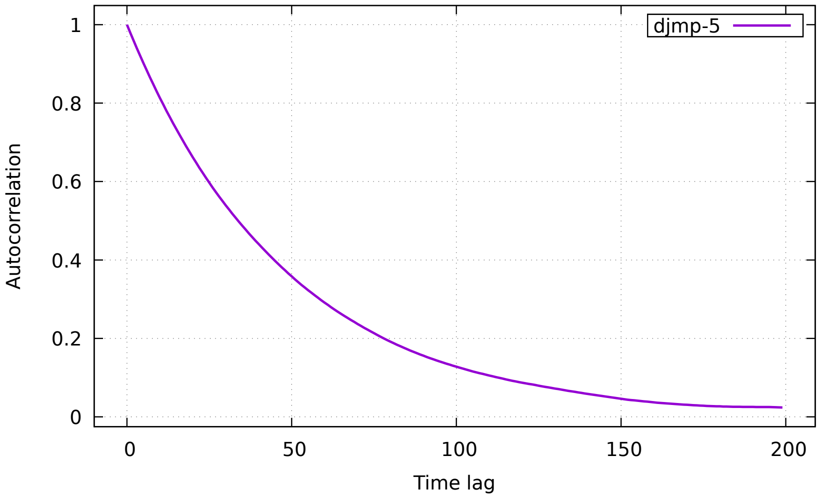
7 jmp-5



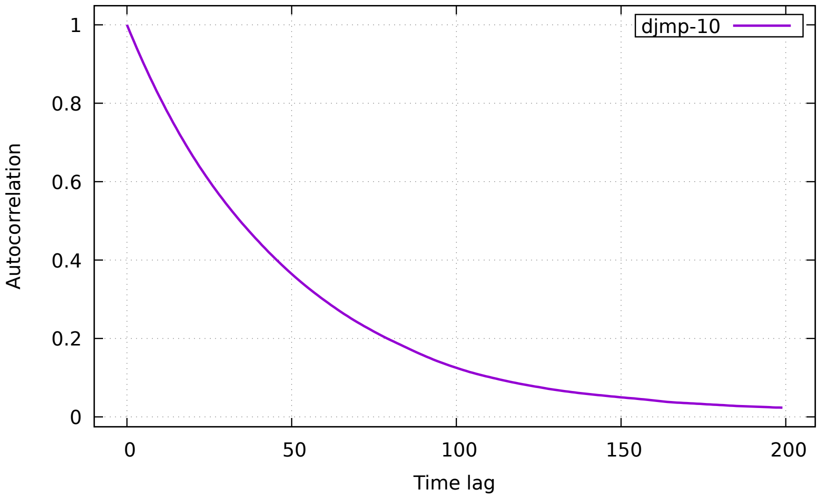
8 jmp-10



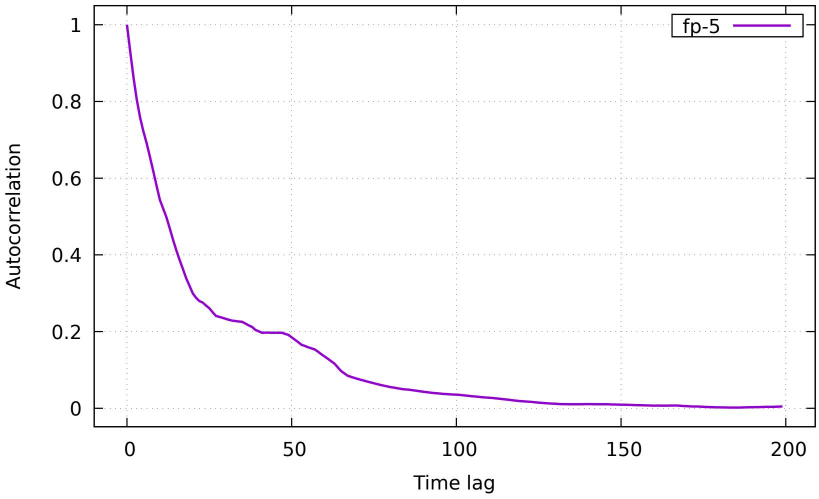
9 djmp-5



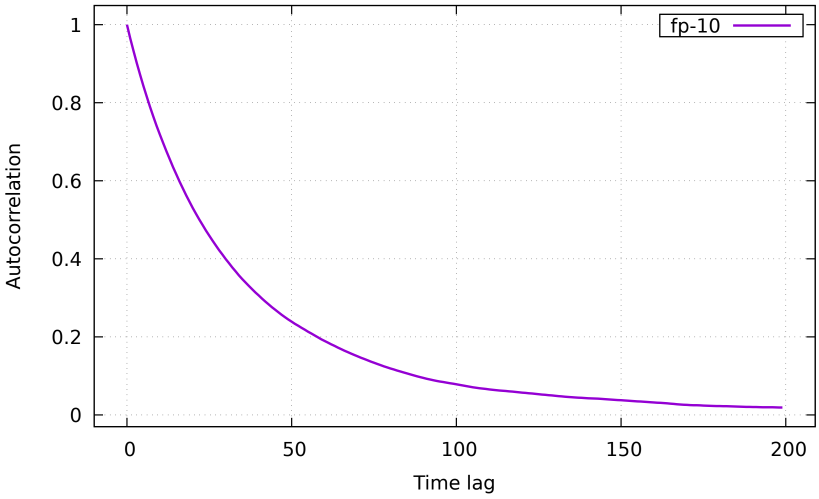
10 djmp-10



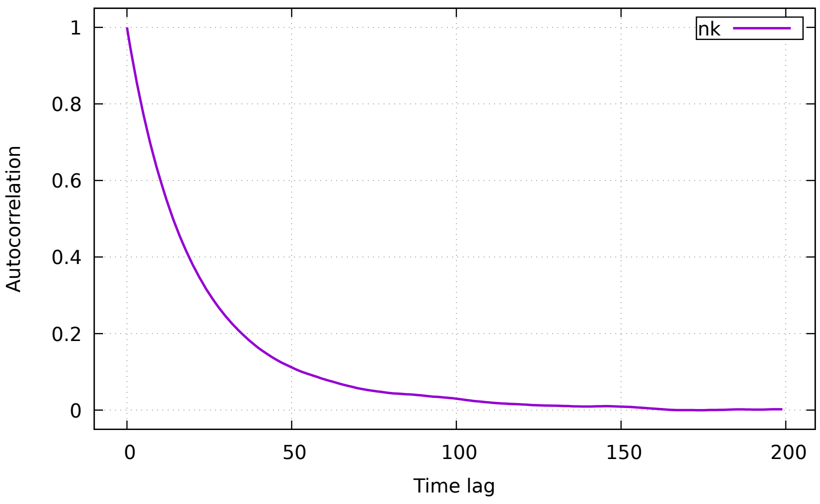
11 fp-5



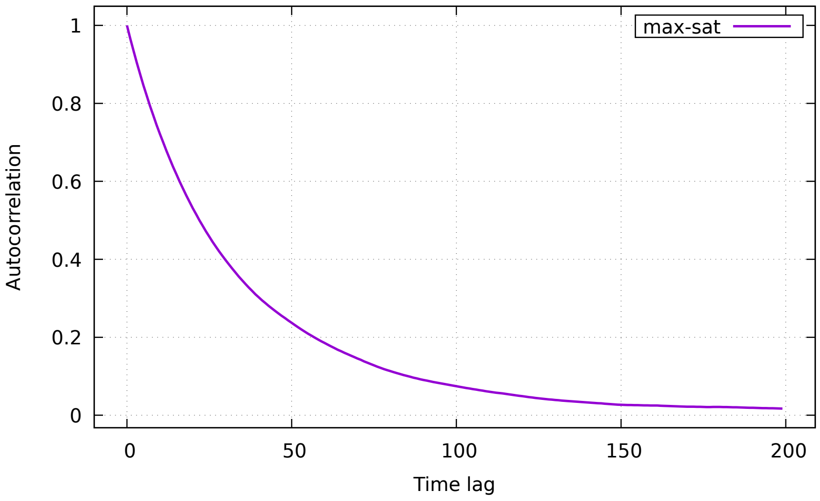
12 fp-10



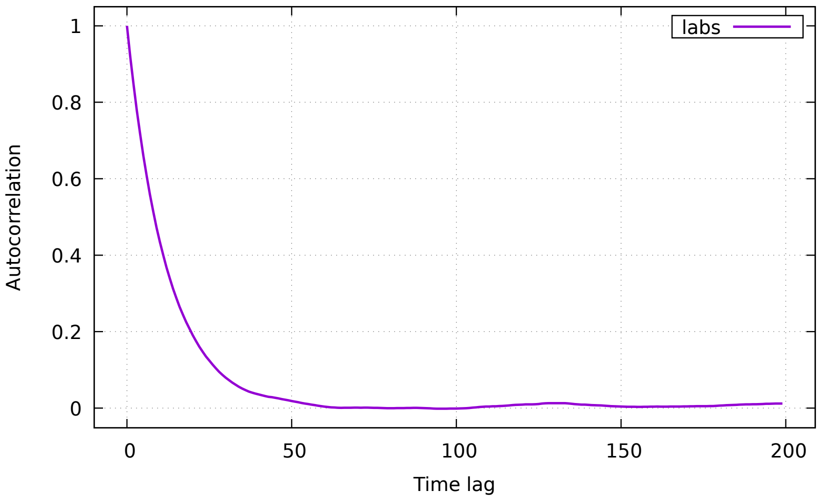
13 nk



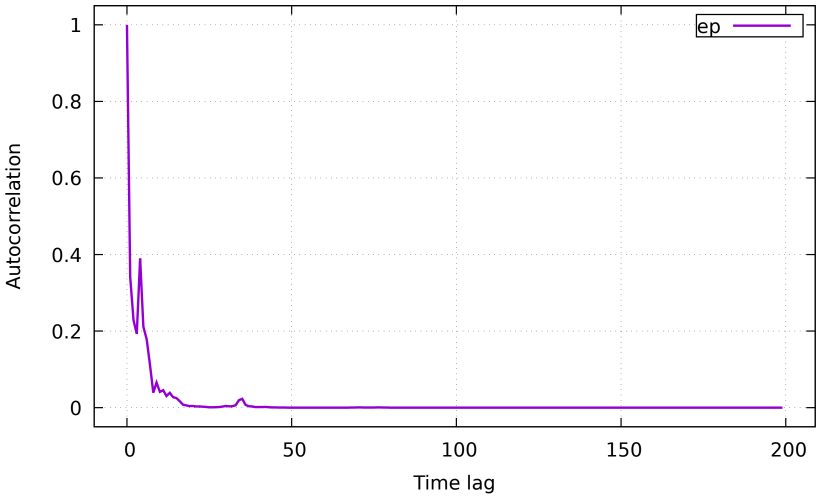
14 max-sat



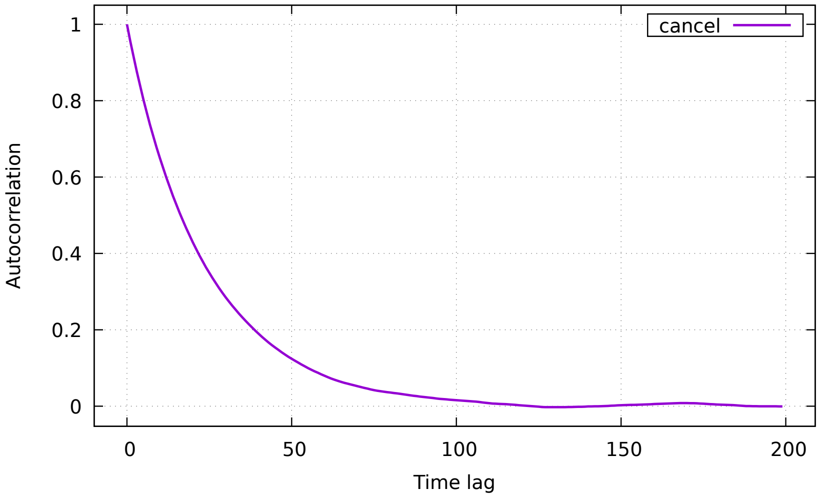
15 labs



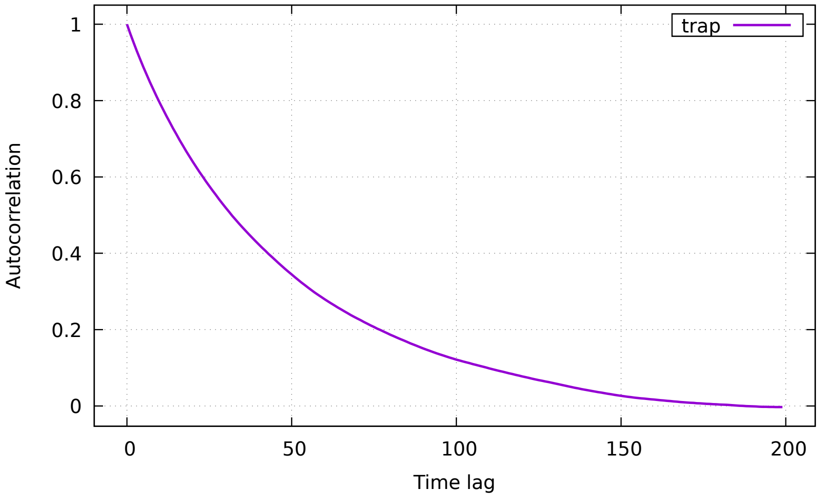
16 ep



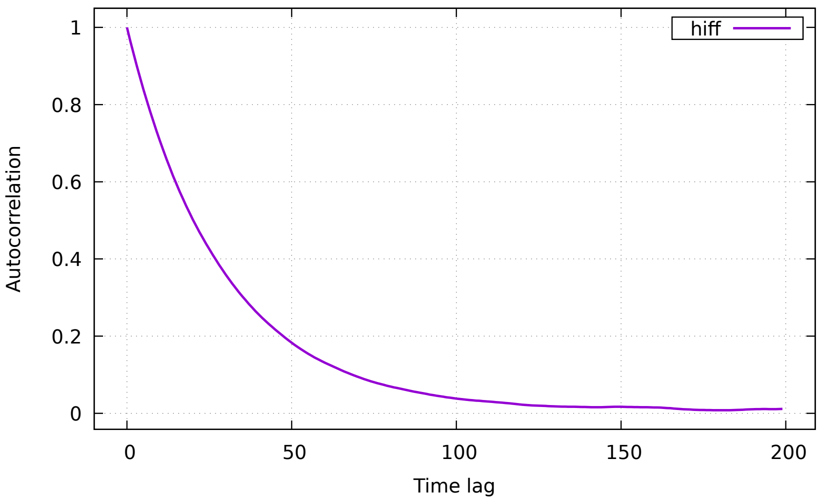
17 cancel



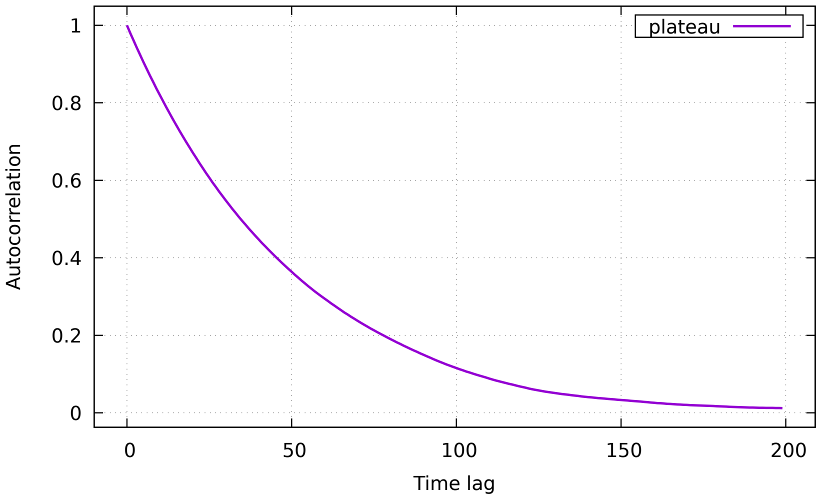
18 trap



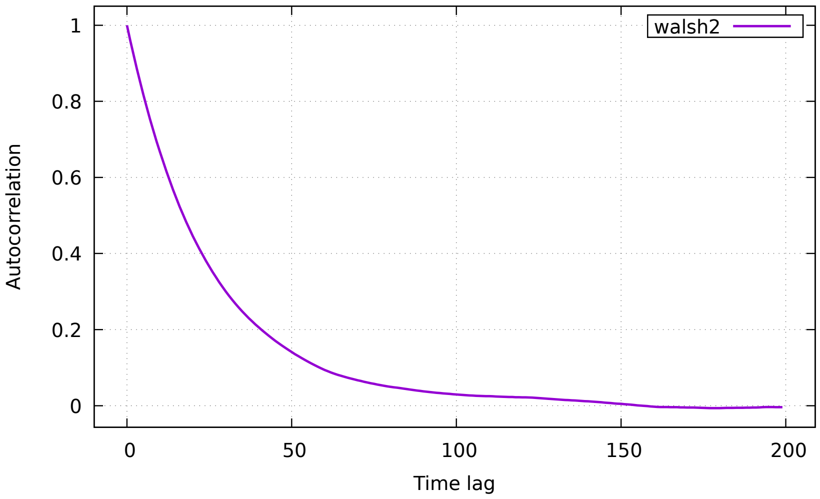
19 hiff



20 plateau



21 walsh2



A Plan

```
{
    "exec": "hnco",
    "opt": "-s 100 -i 500000 -b 0 -A 20 --rw-log-value",
    "parallel": true,
    "results": "results",
    "graphics": "graphics",
    "report": "report",
    "lag_max": 200,
    "functions": [
        {
            "id": "one-max",
            "opt": "-F 0"
        },
        {
            "id": "lin",
            "opt": "-F 1 -p instances/lin.100"
        },
        {
            "id": "leading-ones",
            "opt": "-F 10"
        },
        {
            "id": "ridge",
            "opt": "-F 11"
        },
        {
            "id": "jmp-5",
            "opt": "-F 30 -t 5"
        },
        {
            "id": "jmp-10",
            "opt": "-F 30 -t 10"
        },
        {
            "id": "djmp-5",
            "opt": "-F 31 -t 5"
        },
        {
            "id": "djmp-10",
            "opt": "-F 31 -t 10"
        },
        {
            "id": "fp-5",
            "opt": "-F 40 -t 5"
        },
        {
            "id": "fp-10",
            "opt": "-F 40 -t 10"
        },
        {
            "id": "nk",
            "opt": "-F 60 -p instances/nk.100.4"
        },
        {
            "id": "max-sat",
            "opt": "-F 70 -p instances/ms.100.3.1000"
        },
        {
            "id": "labs",
            "opt": "-F 80"
        },
        {
            "id": "ep",
            "opt": "-F 90 -p instances/ep.100"
        },
        {
            "id": "cancel",
            "opt": "-F 100 -s 99"
        },
        {
            "id": "trap",
            "opt": "-F 110 --fn-num-traps 10"
        },
        {
            "id": "hiff",
            "opt": "-F 120 -s 128"
        },
        {
            "id": "plateau",
            "opt": "-F 130"
        },
        {
            "id": "walsh2",
            "opt": "-F 162 -p instances/walsh2.100"
        }
    ]
}
```

B Default parameters

```
# algorithm = 100
# bm_mc_reset_strategy = 1
# bm_num_gs_cycles = 1
# bm_num_gs_steps = 100
# bm_sampling = 1
# budget = 10000
# bv_size = 100
# description_path = description.txt
# ea_lambda = 100
# ea_mu = 10
# expression = x
# fn_name = noname
# fn_num_traps = 10
# fn_prefix_length = 2
# fn_threshold = 10
# function = 0
# ga_crossover_bias = 0.5
# ga_crossover_probability = 0.5
# ga_tournament_size = 10
# hea_bit_herding = 0
# hea_num_seq_updates = 100
# hea_reset_period = 0
# hea_sampling_method = 0
# hea_weight = 1
# learning_rate = 0.001
# map = 0
# map_input_size = 100
# map_path = map.txt
# map_ts_length = 10
# map_ts_sampling_mode = 0
# mutation_probability = 1
# neighborhood = 0
# neighborhood_iterator = 0
# noise_stddev = 1
# num_iterations = 0
# num_threads = 1
# path = function.txt
# pn_mutation_probability = 1
# pn_neighborhood = 0
# pn_radius = 2
# population_size = 10
# pv_log_num_components = 5
# radius = 2
# real_expression = (1-x)^2+100*(y-x^2)^2
# real_lower_bound = -2
# real_num_bits = 8
# real_upper_bound = 2
# results_path = results.json
# rls_patience = 50
# sa_beta_ratio = 1.2
# sa_initial_acceptance_probability = 0.6
# sa_num_transitions = 50
# sa_num_trials = 100
# seed = 0
# selection_size = 1
# solution_path = solution.txt
# target = 100
# print_defaults
# last_parameter
# exec_name = hnco
# version = 0.14
# Generated from hnco.json
```