HEURISTIC OPTIMIZATION

Empirical Analysis of SLS Algorithms

adapted and extended from slides for SLS:FA, Chapter 4

Analysis of SLS algorithms

- How long does it take to find a feasible solution?
- How good are the solutions generated by the algorithm?
- Which neighborhood structure should I choose?
- How robust is this heuristic algorithm w.r.t. different instances?
- How robust is this heuristic algorithm w.r.t. different parameter settings?
- How does the algorithm scale to large instance size?
- Which is the best of these 4 heuristic algorithms?

- ideally: do theoretical analysis, but:
- the theoretical analysis of detailed behavior of (advanced) SLS algorithms is often very difficult due to
  - stochasticity of algorithms
  - complexity of problems tackled ($\mathcal{NP}$-hard)
  - many degrees of freedom in algorithms which make them theoretically hard to capture
- practical applicability of theoretical results often limited
  - rely on idealised assumptions that do not apply in practical situations (*e.g.*, convergence results for Simulated Annealing)
  - capture only *asymptotic behaviour* and do not reflect *actual behaviour* with sufficient accuracy
  - apply to *worst-case* or *highly idealised average-case*

Therefore:

Analyse the behaviour of SLS algorithms using sound *empirical methodologies*

Example: follow a scientific procedure:

- make observations
- formulate hypothesis/hypotheses (*model*)
- While not satisfied with model (and deadline not exceeded):
  1. design *computational experiment* to test model
  2. conduct computational experiment
  3. analyse experimental results
  4. revise model based on results

Goals of empirical analysis:

Obtain insights into algorithmic performance

- help assessing suitability for applications
- provide basis for comparing algorithms
- characterise algorithm behavior
- facilitate improvements of algorithms

Examine aspects of stochastic search performance

- variability due to randomization
- robustness w.r.t. parameter settings
- robustness across different instances / instance types
- scaling with instance size

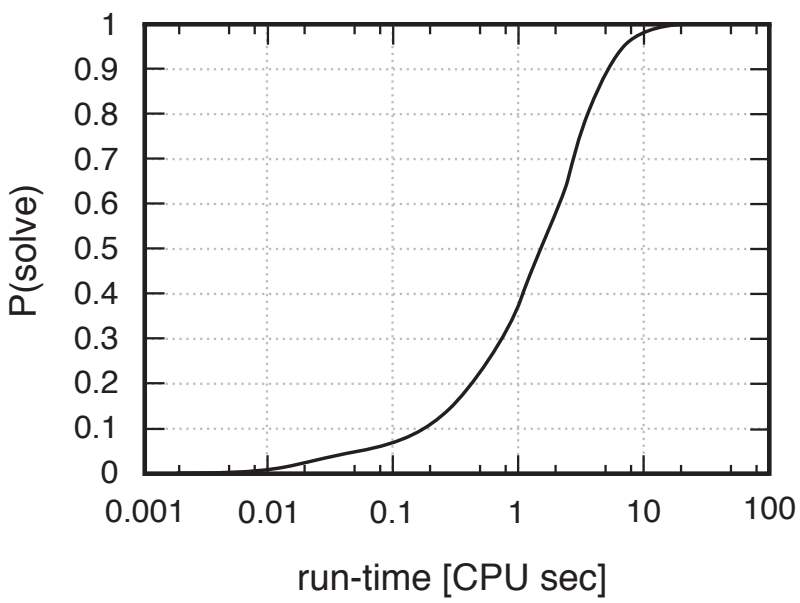
Run-Time Distributions

Empirical analysis of SLS algorithms

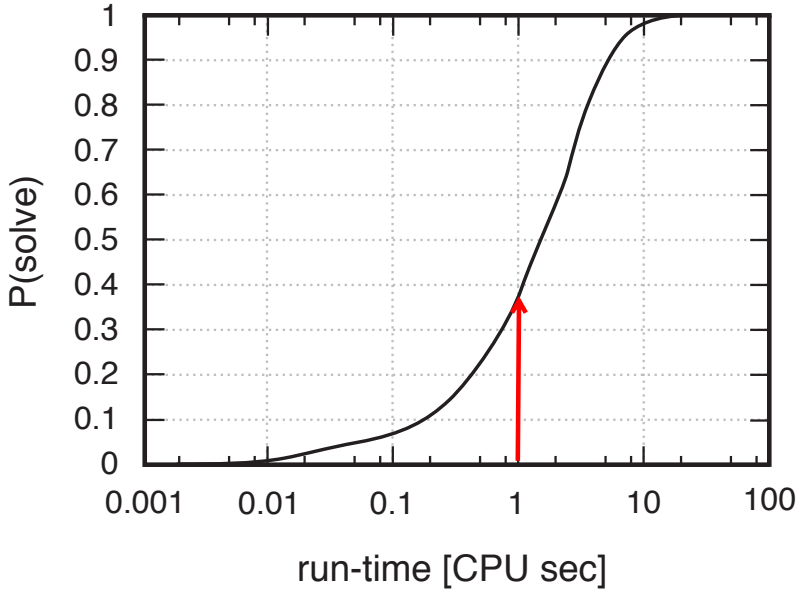
- traditional approach
  - run SLS algorithm n times with time limit t_{max}
  - compute averages, standard deviations
- more detailed point of view
  - for decision SLS algorithms the run time to reach a solution is a random variable
    $\rightsquigarrow$ (univariate) run-time distributions (RTDs)
  - for optimisation SLS algorithms the run time and the solution quality returned are random variables
    $\rightsquigarrow$ (bivariate) run-time distributions (RTDs)

$\rightsquigarrow$ study distributions of *random variables* characterising run-time and solution quality of algorithm on given problem instance.

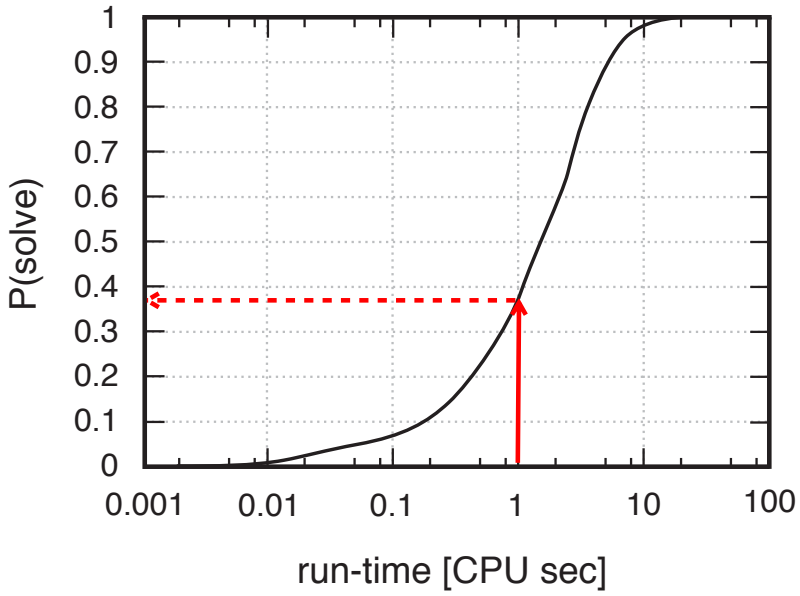
Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial decision problem:



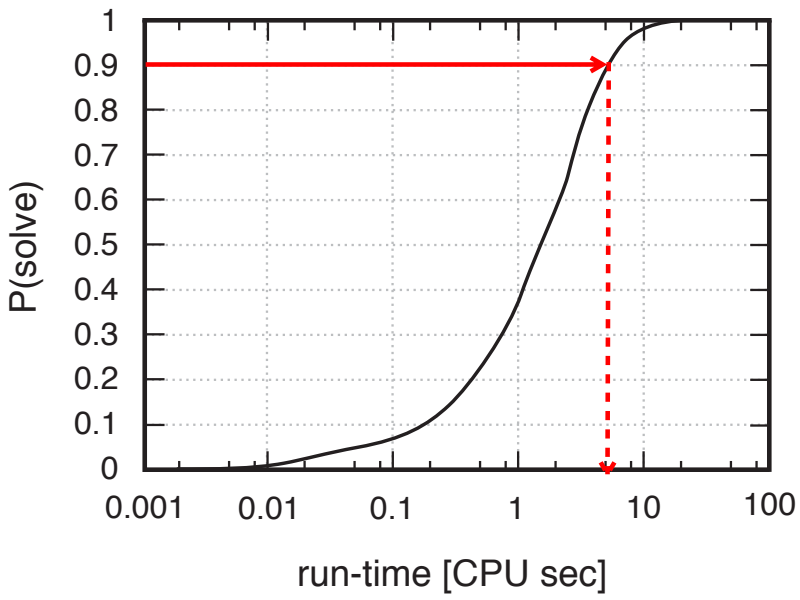
Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial decision problem:



Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial decision problem:



Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial decision problem:



Definition: **Run-Time Distribution (1)**

Given SLS algorithm A for decision problem Π:

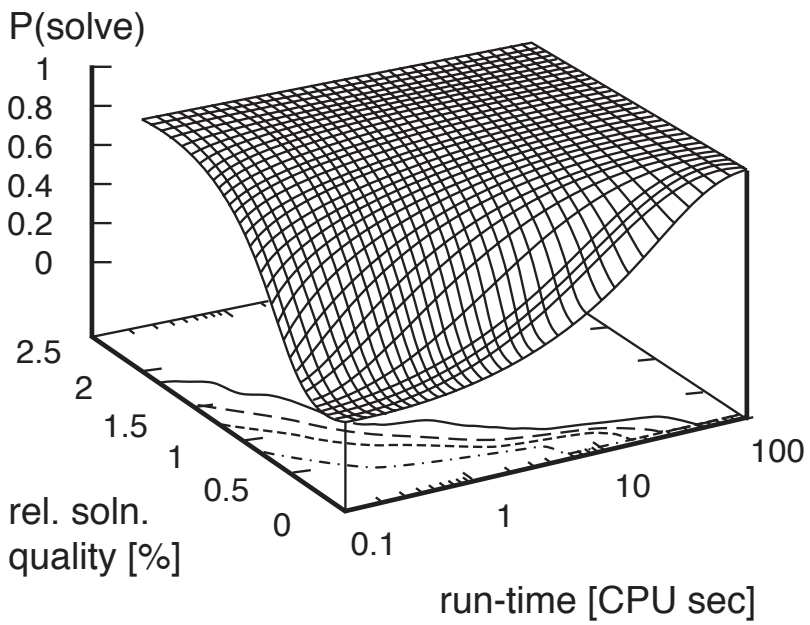
- The *success probability* $P_s(RT_{A,\pi} \leq t)$ is the probability that A finds a solution for a soluble instance $\pi \in \Pi$ in time $\leq t$.
- The *run-time distribution (RTD) of A on π* is the probability distribution of the random variable $RT_{A,\pi}$.
- The *run-time distribution function* $rtd : \mathbb{R}^+ \mapsto [0, 1]$, defined as $rtd(t) = P_s(RT_{A,\pi} \leq t)$, completely characterises the RTD of A on π.

Definition: **Run-Time Distribution (2)**

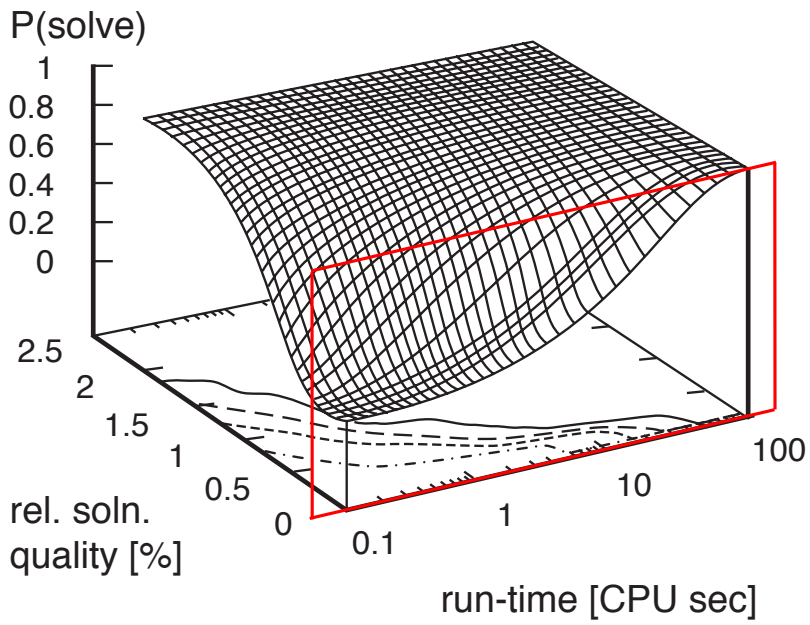
Given SLS algorithm A' for optimisation problem Π':

- The *success probability* $P_s(RT_{A',\pi'} \leq t, SQ_{A',\pi'} \leq q)$ is the probability that A' finds a solution for a soluble instance $\pi' \in \Pi'$ of quality $\leq q$ in time $\leq t$.
- The *run-time distribution (RTD) of A' on π'* is the probability distribution of the bivariate random variable $(RT_{A',\pi'}, SQ_{A',\pi'})$.
- The *run-time distribution function rtd* $: \mathbb{R}^+ \times \mathbb{R}^+ \mapsto [0,1]$, defined as $rtd(t,q) = P_s(RT_{A,\pi} \leq t, SQ_{A',\pi'} \leq q)$, completely characterises the RTD of A' on π'.

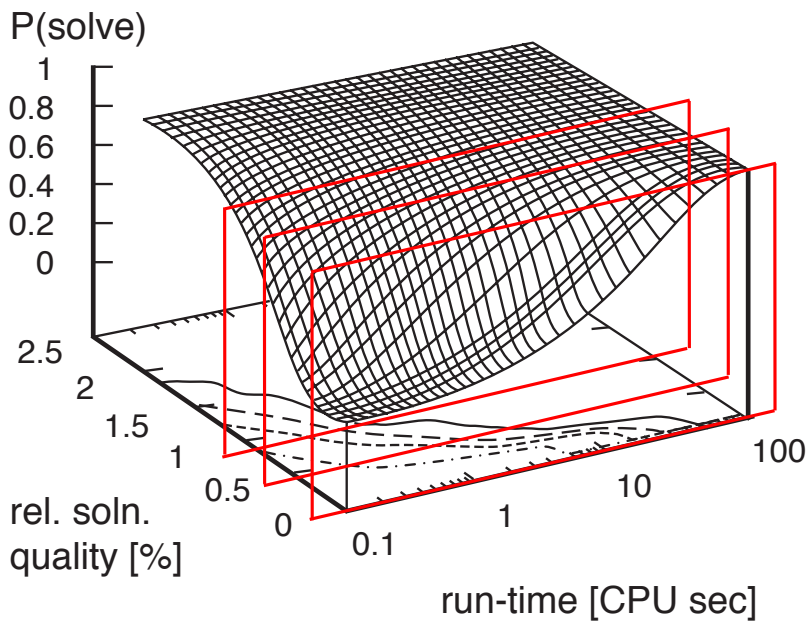
Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial optimisation problem:



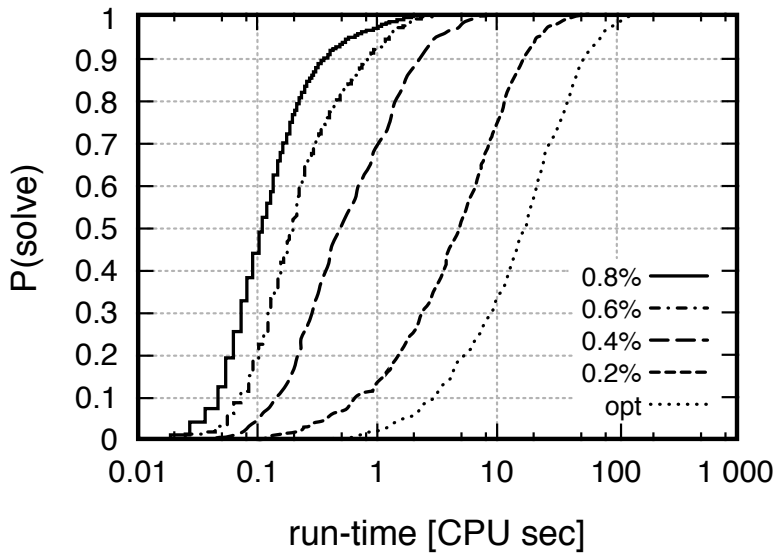
Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial optimisation problem:



Example of run-time distribution for SLS algorithm applied to hard instance of combinatorial optimisation problem:



Qualified RTDs for various solution qualities:



Qualified run-time distributions (QRTDs)

- A *qualified run-time distribution (QRTD) of an SLS algorithm A' applied to a given problem instance π' for solution quality q'* is a marginal distribution of the bivariate RTD $rtd(t, q)$ defined by:

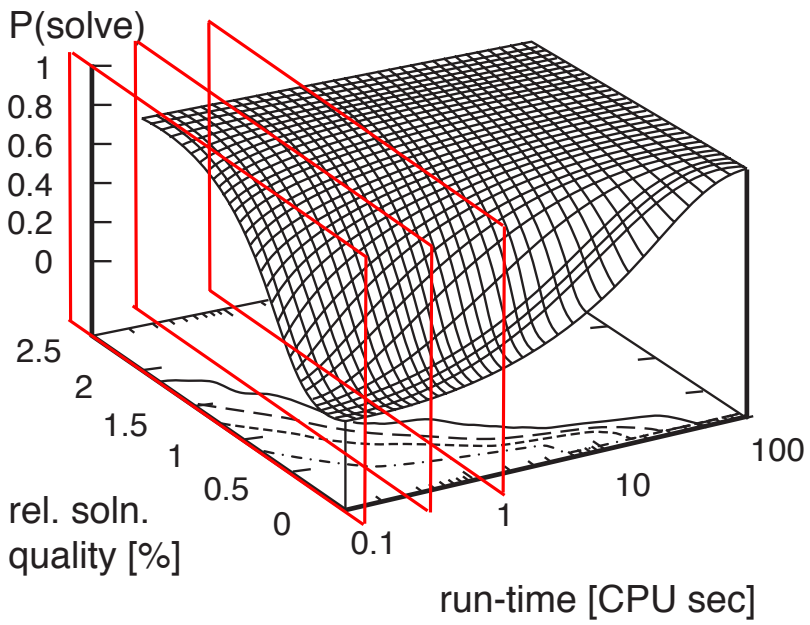
  $$qrtd_{q'}(t) := rtd(t, q') = P_s(RT_{A',\pi'} \leq t, SQ_{A',\pi'} \leq q').$$

- QRTDs correspond to cross-sections of the two-dimensional bivariate RTD graph.

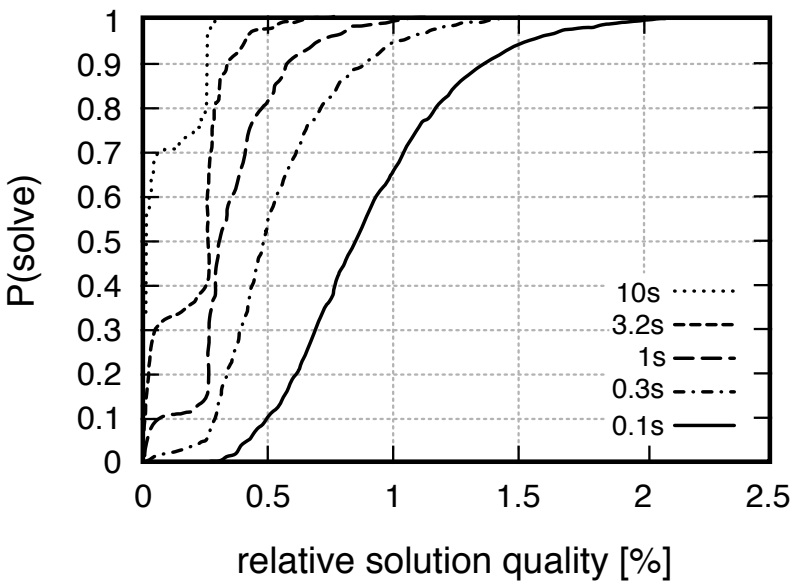
- QRTDs characterise the ability of a given SLS algorithm for a combinatorial optimisation problem to solve the associated decision problems.

Note: Solution qualities q are often expressed as *relative solution qualities* $q/q^* - 1$, where q^* = optimal solution quality for given problem instance.

Typical solution quality distributions for SLS algorithm applied to hard instance of combinatorial optimisation problem:



Solution quality distributions for various run-times:



Solution quality distributions (SQDs)

- A *solution quality distribution (SQD) of an SLS algorithm A' applied to a given problem instance π' for run-time t'* is a marginal distribution of the bivariate RTD $rtd(t, q)$ defined by:

$$sqd_{t'}(q) := rtd(t', q) = P_s(RT_{A',\pi'} \leq t', SQ_{A',\pi'} \leq q).$$

- SQDs correspond to cross-sections of the two-dimensional bivariate RTD graph.

- SQDs characterise the solution qualities achieved by a given SLS algorithm for a combinatorial optimisation problem within a given run-time bound

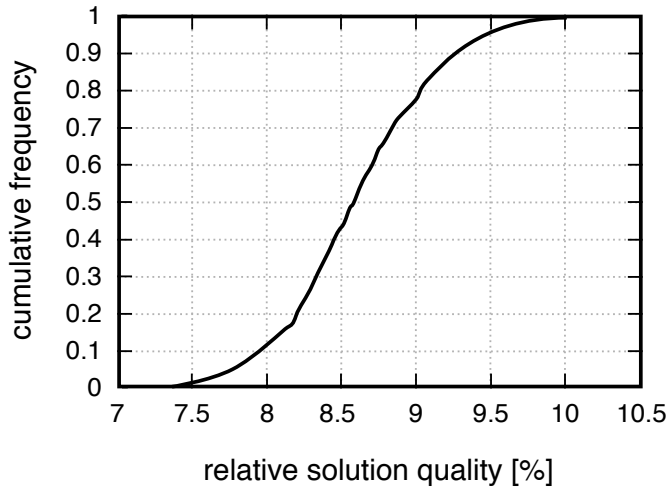
Note:

- For sufficiently long run-times, increase in mean solution quality is often accompanied by decrease in solution quality variability.

- For convergent SLS algorithms, the SQDs for very large time-limits t' approach degenerate distributions that concentrate all probability on the optimal solution quality.

- For any incomplete SLS algorithm A' (such as Iterative Improvement) applied to a problem instance π', the SQDs for sufficiently large time-limits t' approach a non-degenerate distribution called the *asymptotic SQD of A' on π'*.

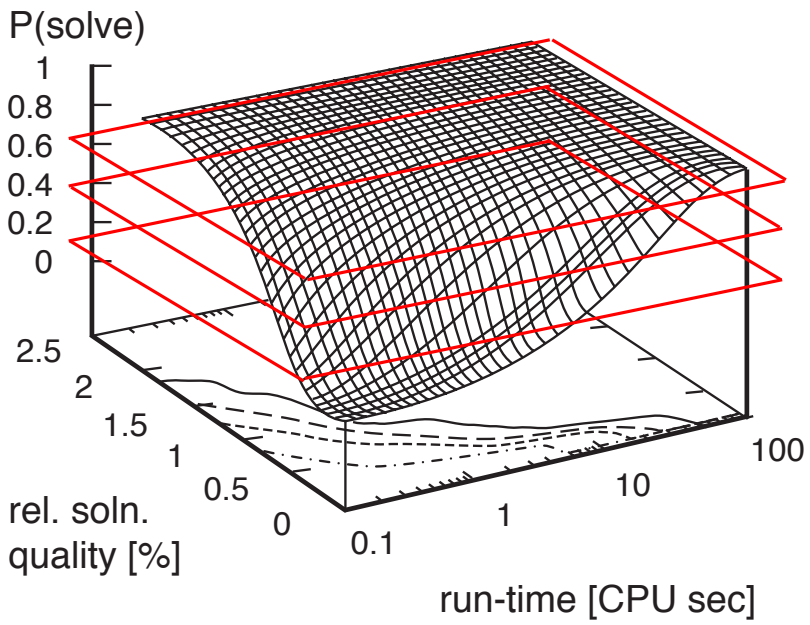
example: asymptotic SQD of Random-order first improvement for TSP instance pcb3038



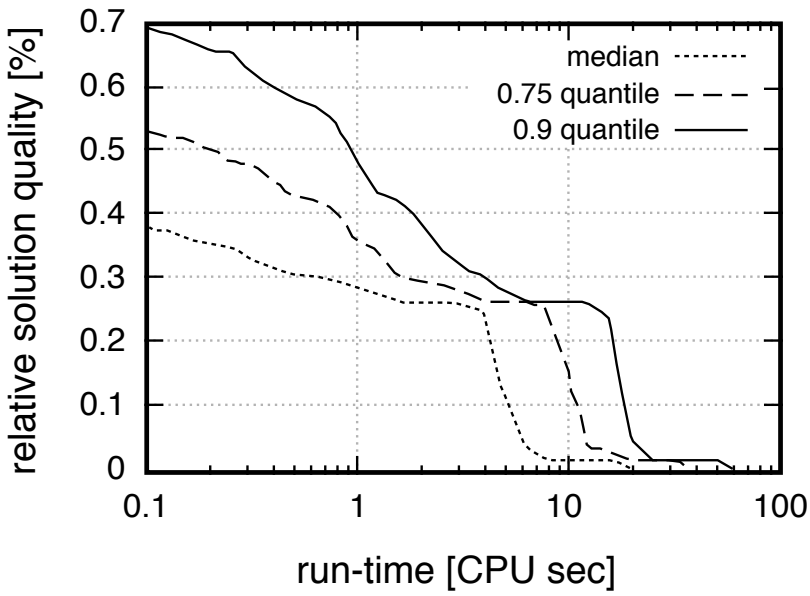
Solution quality statistics over time (SQTs)

- The development of solution quality over the run-time of a given SLS algorithm is reflected in time-dependent SQD statistics (*solution quality over time (SQT) curves*).
- SQT curves are widely used to illustrate the trade-off between run-time and solution quality for a given SLS algorithm.
- SQT curves based on SQD quantiles (such as median solution quality) correspond to contour lines of the two-dimensional bivariate RTD graph.
- **But:** Important aspects of an algorithm's run-time behaviour may be easily missed when basing an analysis solely on a single SQT curve.

Typical SQT curves for SLS optimisation algorithms applied to instance of hard combinatorial optimisation problem:



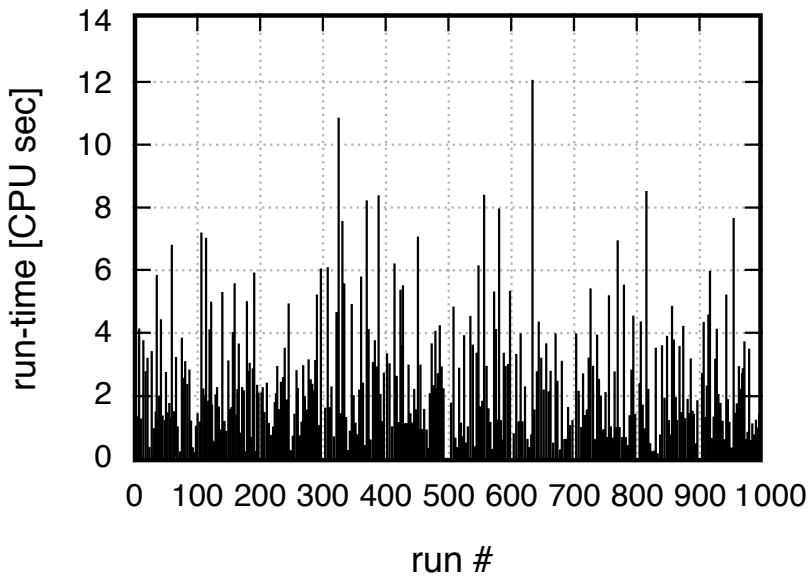
Typical SQT curves for SLS optimisation algorithms applied to instance of hard combinatorial optimisation problem:



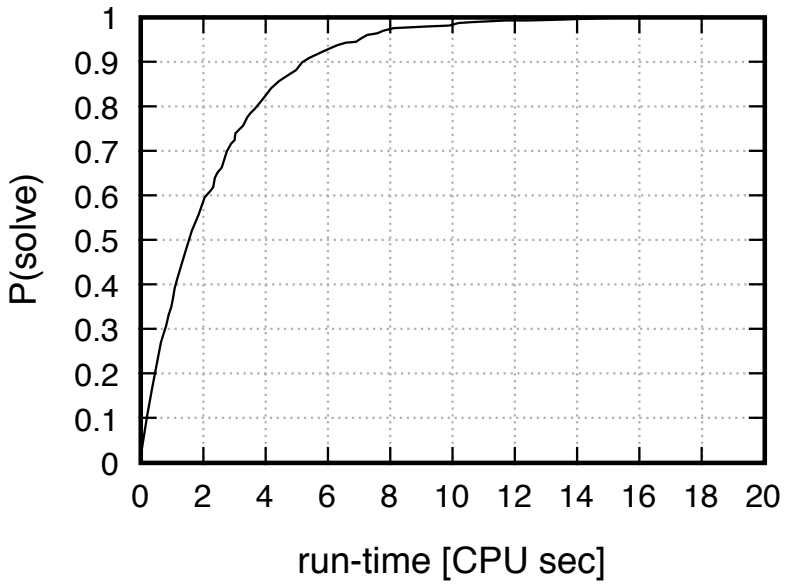
Empirically measuring RTDs

- Except for very simple algorithms, where they can be derived analytically, RTDs are measured empirically.
- Empirical RTDs are approximations of an algorithm's true RTD.
- Empirical RTDs are determined from a number of independent, successful runs of the algorithm on a given problem instance (*samples of true RTD*).
- Higher numbers of runs (larger *sample sizes*) give more accurate approximations of a true RTD.

Typical sample of run-times for an SLS algorithm applied to an instance of a hard decision problem:



Corresponding empirical RTD:



Protocol for obtaining the empirical RTD for an SLS algorithm A applied to a given instance π of a decision problem:

- Perform k independent runs of A on π with cutoff time t'. (For most purposes, k should be at least 50–100 and t' should be high enough to obtain at least a large fraction of successful runs.)
- Record number k' of successful runs, and for each run, record its run-time in a list L.
- Sort L according to increasing run-time; let $rt(j)$ denote the run-time from entry j of the sorted list ($j = 1, \ldots, k'$).
- Plot the graph $(rt(j), j/k)$, *i.e.*, the cumulative empirical RTD of A on π.

Note:

- The fraction of successful runs, $sr := k'/k$, is called the *success ratio*; for large run-times t', it approximates the *asymptotic success probability* $p_s^* := lim_{t\to\infty} P_s(RT_{a,\pi} \leq t)$.
- In cases where the success ratio sr for a given cutoff time t' is smaller than 1, quantiles up to sr can still be estimated from the respective truncated RTD.

  The mean run-time for a variant of the algorithm that restarts after time t' can be estimated as:

  $$\widehat{E}(RT_s) + (1/sr - 1) \cdot \widehat{E}(RT_f)$$

  where $\widehat{E}(RT_s)$ and $\widehat{E}(RT_f)$ are the average times of successful and failed runs, respectively.

  Note: $1/sr - 1$ is the expected number of failed runs required before a successful run is observed.

Protocol for obtaining the empirical RTD for an SLS algorithm A' applied to a given instance π' of an optimisation problem:

- Perform k independent runs of A' on π' with cutoff time t'.
- During each run, whenever the incumbent solution is improved, record the quality of the improved incumbent solution and the time at which the improvement was achieved in a *solution quality trace*.
- Let $sq(t', j)$ denote the best solution quality encountered in run j up to time t'. The cumulative empirical RTD of A' on π' is defined by $\widehat{P}_s(RT \leq t', SQ \leq q') := \#\{j \mid sq(t', j) \leq q'\}/k$.

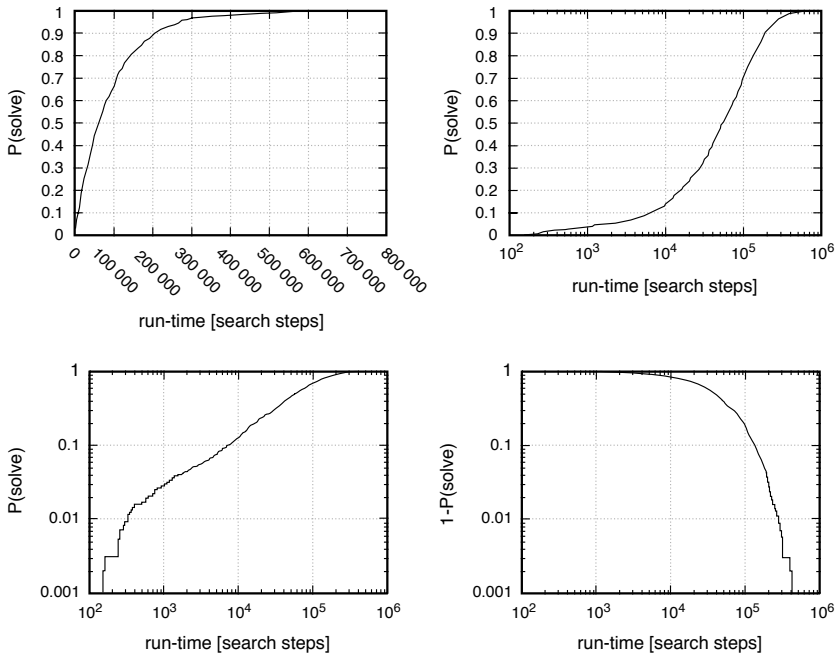
Note: Qualified RTDs, SQDs and SQT curves can be easily derived from the same solution quality traces.

```
Parameter-settings:
max-tries 5
time 10.000000
num-ants 25
num-neigh 20
alpha 1.000000
beta 2.000000
rho 0.500000
q_0 0.000000
ls_flag 3
nn_ls 20
dlb_flag 1
mmas_flag 1
begin problem /Users/stuetzle/Benchmarks/TSP/lin318.tsp
seed 1335270635
begin try 0
best 42814 iteration 1 tours 27 time 0.020
best 42775 iteration 3 tours 77 time 0.050
best 42711 iteration 4 tours 102 time 0.060
best 42582 iteration 7 tours 177 time 0.110
best 42362 iteration 8 tours 202 time 0.120
best 42253 iteration 10 tours 252 time 0.140
best 42199 iteration 11 tours 277 time 0.150
best 42181 iteration 26 tours 652 time 0.280
best 42159 iteration 30 tours 752 time 0.320
best 42143 iteration 34 tours 852 time 0.360
best 42101 iteration 1021 tours 25527 time 9.120
best 42080 iteration 1024 tours 25602 time 9.160
best 42050 iteration 1025 tours 25627 time 9.170
best 42029 iteration 1027 tours 25677 time 9.190
end try 0
seed 1528632406
begin try 1
best 42747 iteration 1 tours 27 time 0.020
best 42681 iteration 4 tours 102 time 0.060
```


Different views of RTD plots are useful for the *qualitative analysis* of SLS behaviour:

- *Semi-log plots* give a better view of the distribution over its entire range.
- Uniform performance differences characterised by a constant factor correspond to shifts along horizontal axis.
- *Log-log plots* of an RTD or its associated *failure rate decay function*, $1 - rtd(t)$, are often useful for examining behaviour for very short or very long runs.

Various graphical representations of a typical RTD:



Measuring run-times (1):

- CPU time measurements are based on a specific *implementation* and *run-time environment* (machine, operating system) of the given algorithm.

- To ensure reproducibility and comparability of empirical results, CPU times should be measured in a way that is as independent as possible from machine load.

  When reporting CPU times, the run-time environment should be specified (at least CPU type, model, speed and cache size; amount of RAM; OS type and version); ideally, the implementation of the algorithm should be made available.

Measuring run-times (2):

To achieve better abstraction from the implementation and run-time environment, it is often preferable to measure run-time using

- *operation counts* that reflect the number of operations that contribute significantly towards an algorithms performance, and
- *cost models* that specify the CPU time for each such operation for a given implementation and run-time environment.

Example:

For a given SLS algorithm for SAT applied to a specific SAT instance we observe

- a median run-time of 38 911 search steps (*operation count*);
- the CPU time required for each search step is 0.027ms, while initialisation takes 0.8ms (*cost model*)

  when running the algorithm on an Intel Xeon 2.4GHz CPU with 512KB cache and 1GB RAM running Red Hat Linux, Version 2.4smp (*run-time environment*).