MATH 3172 3.0
Combinatorial Optimization

Midterm I

Jacques Nel

February 17, 2020

1 Hill climb

1.1 `grid1` and `grid2`

$$\texttt{grid1} = \begin{bmatrix} 3 & 7 & 2 & 8 \\ 5 & 2 & 9 & 1 \\ 5 & 3 & 3 & 1 \end{bmatrix}, \qquad \text{and} \qquad \texttt{grid2} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 2 & 8 & 10 \\ 0 & 2 & 4 & 8 & 16 \\ 1 & 4 & 8 & 16 & 32 \end{bmatrix}.$$

The global maximum of `grid1` and `grid2` was trivial to find using a naive implementation of the hill-climb. Both adjacent and diagonal state transitions were allowed to reduce the number of iterations.

Table 1: Hill-climb for three given discrete functions

Function $f(x)$	iterations	time (μs)	$x^\star$	$f(x^\star)$	success
`grid1`	3	112	$(1, 2)$	9	yes
`grid2`	2	39	$(3, 4)$	32	yes
`grid3`	8	115	$(7, 98)$	-7.4	no

1.2 `grid3` is problematic

Observe in the table, in the previous section, the naive hill climbing algorithm fails to find the global maximum of `grid3` around the point $x^\star = (1, 1)$.

Figure 1: 3D plot of $f_3(\boldsymbol{x})$

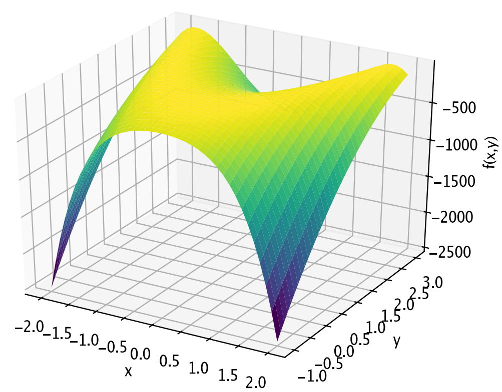
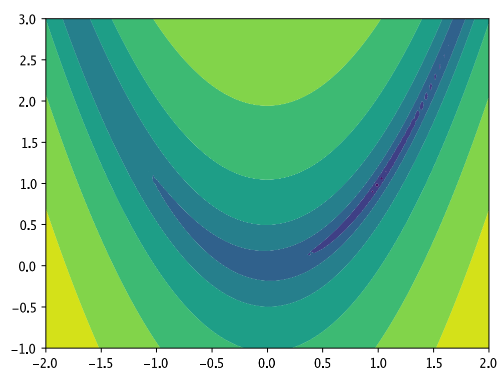


Figure 2: $(-f_3(\boldsymbol{x}))^{1/8}$ to emphasize narrow global maximum band



Especially when discretized, `grid3` has a ridge on which $\boldsymbol{x}^\star = (1, 1)$ lies. Hill climbing tends to get stuck along the sides this ridge, causing the algorithm to fail to find the global maximum. Aliasing, as a result of discretization, also results in several small isolated maxima near this ridge in the vicinity of $\boldsymbol{x}^\star$.

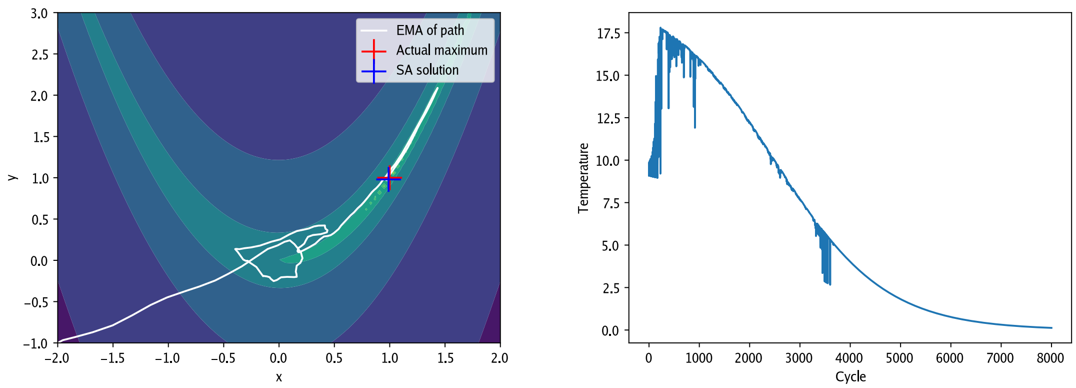
2 Simulated annealing

2.1 grid3

2.1.1 Results

Figure 3: Simulated annealing with adaptive exponential cooling schedule

(a) State trajectory (b) Adaptive-AE cooling schedule



An adaptive additive exponential cooling schedule was used to find the global maximum of `grid3`. Due to the large number of steps, the white line denotes an exponential moving average or EMA of the trajectory with $\gamma = 0.001$, instead of showing the whole state trajectory.

$$T_k = T_n + (T_0 - T_n)\left(\frac{1}{1 + e^{\frac{2\ln(T_0 - T_n)}{n}(k - \frac{1}{2}n)}}\right) \tag{1}$$

Furthermore, we multiply T_k by an adaptive term $1 \leq \mu \leq 2$ which is calculated using the distance between the value of the current state $f(s_i)$ and the best value encountered so far $f^\star$.

$$T = \mu T_k = \left(1 + \frac{f(s_i) - f^\star}{f(s_i)}\right) T_k \tag{2}$$

In practice, we take the `np.abs` and use `np.clip(x, 1, 2)` to ensure that these assumptions are maintained.

2.1.2 Other cooling schedules considered

The following monotonic additive and multiplicative cooling schedules were also tried, but failed to produce good results:

1. Linear cooling,
2. (a) exponential multiplicative cooling,
   (b) logarithmic multiplicative cooling,
   (c) quadratic multiplicative cooling,
3. (a) linear additive cooling,
   (b) quadratic aditive cooling,
   (c) exponential additive cooling.