† **Electronic Supplementary Information (ESI) available:** [details of any supplementary information available should be included here]. See DOI: 10.1039/b000000x/

1 Searching of global minimum

The objective function to be minimized is written as Eq. (17) of the main text (here Eq. (S1)),

$$A(0)/V(0) = \sum_{\gamma=1}^{Q} X(\gamma) A\left(\{\rho_1^{(\mathrm{p})}(\gamma), \rho_2^{(\mathrm{p})}(\gamma)\}, T\right) \Big/ V(\gamma). \tag{S1}$$

Additional restrictions are:

$$1 = \sum_{\gamma=1}^{Q} X(\gamma), \tag{S2}$$

$$\rho_{\mathrm{i}}^{(\mathrm{p})}(0) = \sum_{\gamma=1}^{Q} X(\gamma) \rho_{\mathrm{i}}^{(\mathrm{p})}(\gamma). \tag{S3}$$

Insertion of the restrictions (S2) and (S3) into the objective function, Eq. (S1), reduces the number of independent variables and leads to the formal expression (at overall protein densities $\rho_1^{(\mathrm{p})}(0)$, $\rho_2^{(\mathrm{p})}(0)$ and temperature T)

$$A(0)/V(0) = f\left(X(1), \rho_1^{(\mathrm{p})}(1), \rho_2^{(\mathrm{p})}(1); X(2), \rho_1^{(\mathrm{p})}(2), \rho_2^{(\mathrm{p})}(2) \cdots X(Q-1), \rho_1^{(\mathrm{p})}(Q-1), \rho_2^{(\mathrm{p})}(Q-1)\right). \tag{S4}$$

The volume fraction of the phase (γ) and both protein densities are denoted as a triplet $X(\gamma), \rho_1^{(\mathrm{p})}(\gamma)$, $\rho_2^{(\mathrm{p})}(\gamma)$. All together, $3Q-3$ unknowns must be determined in the case of decomposition into Q phases. Note that the last triplet, $X(Q), \rho_1^{(\mathrm{p})}(Q), \rho_2^{(\mathrm{p})}(Q)$, is determined via Eqs. (S2) and (S3). The system decomposes into Q coexisting phases, if the objective function reaches the global minimum. In a binary mixture, the cases where $Q=2$ and/or $Q=3$ must be considered for each state point $\rho_1^{(\mathrm{p})}(0)$, $\rho_2^{(\mathrm{p})}(0)$, T. There are four possible scenarios:

- no global minimum is found for any Q: the system is stable as one phase,
- global minimum is found only for $Q=2$: coexistence of two phases,
- global minimum is found only for $Q=3$: coexistence of three phases,
- global minimum is found for $Q=2$ and $Q=3$: number of equilibrium phases is determined by the smaller value of the objective function.

Minimization of the objective function for state points $\rho_1^{(\mathrm{p})}(0), \rho_2^{(\mathrm{p})}(0)$ at certain T, allows us to determine the volume fractions and the densities of equilibrium phases $X(\gamma), \rho_1^{(\mathrm{p})}(\gamma), \rho_2^{(\mathrm{p})}(\gamma)$, for $\gamma = 1 \cdots Q$. For this purpose we used the Boltzmann simplex simulated annealing (BSSA) method. The origin of this method relies on the Nelder–Mead method, known also as the downhill simplex method.[1] First we randomly chose (if we search for a coexistence of Q phases) $Q+1$ initial approximations for a global minimum. An approximation is a set of $Q-1$ triplets, see also the argument of f in Eq. (S4). We proceed toward global minimum by construction of simplex objects in a hyperplane of approximations, using criteria for the steepest descent. Unfortunately, such an approach may lead to a local minimum which, may be far away from the global one. To solve this difficulty, we have to ensure a non–zero probability for passing over the free energy barrier, reaching the global minimum. This is done by the BSSA method as the upgrade of the original Nelder–Mead method. Unlike criteria of the steepest descent (downhill) we allow for approximations to reach higher free energy values (uphill) with the Boltzmann probability $p(E) = \exp(-E/T_a)$, where $E = A(0)/V(0)$ and T_a the “annealing temperature”. The uphill step is implemented by the traditional Metropolis algorithm.[2] The similarity with the statistical mechanical methods is coincidental: note that the BSSA method may be used to minimize not only energy functionals but also the others. In the latter cases, T_a should be interpreted as a control parameter, which regulates the uphill–downhill step ratio.

A simplified pseudo code, used in our work, is shown below:

```
# search the equilibrium phases for a set of temperatures
T=1...

  # scanning of a density sets of state poins
  rho1_p=1...
  rho2_p=1...

    # overall protein densities and temperature (state points) are now determined

    # find two- or three coexistence region
    Q=2,3

      # determine Q+1 initial approximations x and their objective functions A0_V0
      i=1,Q+1
      x(i)=rand
      A0_V0(i)=f(x(i))

        # until mismatch of equilibrium conditions (Eqs. (13) and (14) of the main text) is above 0.01%
        WHILE(equilibrium conditions)
          USE BSSA method
          CHECK equilibrium conditions

    DETERMINE one-, two-, or three phase region according to four possible scenarios
```

Please note the following:

1. the annealing procedure within BSSA method must be adjusted to the appropriate scale, for each binary mixture separately,

2. triplet $X(\gamma), \rho_1^{(\mathrm{p})}(\gamma), \rho_2^{(\mathrm{p})}(\gamma)$ must be non–negative and such that its packing fraction, $\eta = \pi\sigma^3[\rho_1^{(\mathrm{p})}(\gamma) + \rho_2^{(\mathrm{p})}(\gamma)]/3$, does not exceed the value ($\pi\sqrt{2}/6 \approx 0.74$), within each phase ($\gamma$). If this happens, new approximation must be generated,

3. BSSA is not the universal method for minimization, but it turned out to be efficient enough for our purposes. Reader is referred to ref. 1 (chapters 10.4 and 10.9) for details of implementation.

2 Cloud and shadow curves

Cloud and shadow curves offer an alternative description of the two phase equilibrium. To define the terminology, we assume that we are within the one–phase region (mother phase) at $T > T_{\mathrm{cloud}}$, for given packing fraction and composition. As T is lowered and it reaches $T = T_{\mathrm{cloud}}$ (cloud point), the mother phase coexists with an infinitesimal amount of the second phase, called the “shadow” phase. The packing fraction and composition of the shadow phase differ from those of the mother phase. To obtain the cloud and shadow curves we plot the collection of mother phases at cloud point temperatures at different packing fraction of constant composition (for a cloud curve, see also Fig. 4 of the main text), and the coexisting shadow phases with their packing fractions and compositions (shadow curve). Cloud and shadow curves coincide for the one–component systems, see panel A of Fig. 4. The case of γD–βB1 mixtures is presented in Fig. S1. We present the cloud curve at $x_2 = 0.08$ shown before (see Fig. 4 of the main text), together with the associated shadow curve. In this way, the two distinct curves describe the temperature dependence of protein partitioning (at overall composition x_2) as one simple 2D plot.

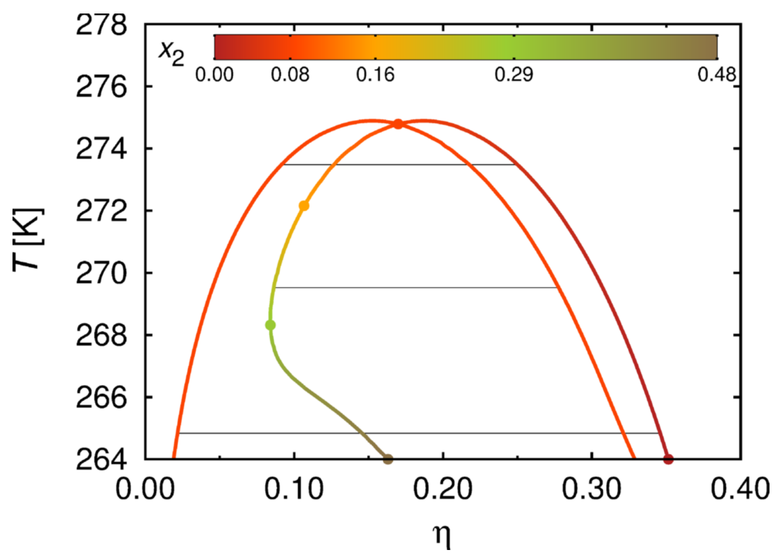

Fig. S1 The cloud curve at $x_2 = 0.08$ (monochrome–orange) and its corresponding shadow curve. Shadow curve assumes for x_2 values to vary continuously from 0.00 to 0.48. Intermediate values are labeled by colored circles as suggested by the color bar on top of the figure. The cloud and shadow curves intersect at x_2 equal to 0.08. The color code is the same as the code for the cloud curves in Fig. 4 of the main text. Black tie–lines connect the coexisting phases.

3 Generating Hammer projection

The way of generating the Hammer projection applied here greatly relies on the paper of Koromyslova[3] and co–workers. Here we sketch only the major differences to ref. 3. To project the charged residues on the protein surface as seen in Fig. 7(b) of the main text, the input file "Surface_charge.dat" is needed. The latter should contain the list of amino acid residues, atomic coordinates, and the labels of amino acids (1–positively charged, 0–negatively charged, 0.5–uncharged). The short section of the input file "Surface_charge.dat" looks like:

```
LYS     2     15      -4.793     -9.509     6.105     1
LYS     2     16      -3.923    -10.896     5.936     1
LYS     2     17      -2.839    -10.192     3.908     1
LYS     2     18      -3.190     -8.493     4.216     1
LYS     2     19      -5.383    -10.574     3.799     1
LYS     2     20      -4.678     -9.671     2.469     1
LYS     2     21      -5.344     -7.523     3.605     1
LYS     2     22      -6.079     -8.440     4.891     1
ARG     9     23      -9.030      7.443     0.137     1
ARG     9     24      -9.565      7.533     1.501     1
ARG     9     25     -10.922      6.826     1.586     1
ARG     9     26     -11.614      6.680     0.577     1
```

The 1st column is the abbreviated name of amino acid residue on the protein surface, the 2nd its position in amino acid sequence, the 3rd column is the cumulative counter of atoms, columns 4, 5, and 6 describe the atomic coordinates within amino acid, and finally column 7 the label of amino acid. Note that lysine (LYS) and arginine (ARG) are positively charged (label 1) at the investigated pH (7.0). The selection of surface residues and their charges can be extracted from PDB file using an appropriate software (Maestro Schödinger, Visual Molecular Dynamics...), and organizing the data into the desired format.

After the input data "Surface_charge.dat" is created, the utilization of the code below (Python language) gives the output as "Surface_charge.pdf". Some of the steps are briefly documented.

```
# All libraries needed
import numpy as np
from scipy.interpolate import griddata
```

```
import matplotlib
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
from matplotlib.colors import LinearSegmentedColormap

# Defining the color pallete "cdict"
cdict = {'red':   ((0.0,0.9,0.9), (0.15,1.0,1.0), (0.15,1,1), (0.85,1,1), (0.85,0.4,0.4), (1.0,0.0,0.0)),
         'green': ((0.0,0.0,0.0), (0.15,0.5,0.5), (0.15,1,1), (0.85,1,1), (0.85,0.5,0.5), (1.0,0.0,0.0)),
         'blue':  ((0.0,0.0,0.0), (0.15,0.0,0.0), (0.15,1,1), (0.85,1,1), (0.85,1.0,1.0), (1.0,0.5,0.5))}

# Assignation of the collor pallete "cdict" to "manual_cbar"
manual_cbar = LinearSegmentedColormap('manual_cbar', cdict)
plt.register_cmap(cmap=manual_cbar)

lons = np.arange(-180, 181, 1)                        # Generating spheric coordinate grid - Longitudes
lats = np.arange(-90, 91, 1)                          # Generating spheric coordinate grid - Latitudes
map = Basemap(projection='hammer', lon_0=0)           # Initializing Hammer projection from the Basemap library
x, y = map(*np.meshgrid(lons, lats))                  # Coordinate grid in the projection of the map

# Reading of the input data (here Surface_charge.dat)
def read_surf(surfFile):
  dots = np.loadtxt(surfFile, usecols = (3, 4, 5))     # Read the x,y,z coordinates of atom within amino acid
  mhp = np.loadtxt(surfFile, usecols = (6, ))          # Assignate the label of these atom:
positive(1)-negative(0)-neutral(0.5)
  return dots, mhp

# Conversion into spheric coordinate system
def spheric(D):
  D_c = D - D.mean(axis = 0)                          # D_c are radius-vectors for surface dots
  c = []
  h = []
  for d in D_c:
    x, y, z, r = d[0], d[1], d[2], np.sqrt((d**2).sum())
    lat = np.arcsin(z / r) * 180 / np.pi              # Latitudes
    lon = np.arctan2(y, x) * 180 / np.pi              # Longitudes
    c.append([lon, lat])                              # Update the list
  return np.array(c), np.array(h)

#Function for building spheric projection maps
def make_map(Surface, Mhp, saveMap = False):
  DS, sH = spheric(Surface)                           # Dots on the sphere
  MhpS = griddata(DS, Mhp, (lons[None, :], lats[:, None]), method = 'cubic')
  if saveMap == True:
    f = plt.figure()
    CS = plt.contourf(x, y, MhpS, levels = np.arange(0, 1.01, 0.01), extend = 'both', cmap = manual_cbar)
    map.drawparallels(np.arange(-60., 90., 30.))
  map.drawmeridians(np.arange(-180, 180, 59.99))
  plt.savefig('Surface_charge.pdf', format = 'pdf')    # Here, the map is saved as PDF figure.
  return MhpS

# Final output as pdf
Dots, Mhp = read_surf('Surface_charge.dat')
MhpS = make_map(Dots, Mhp, True)
```

References

S1 *Numerical Recipes in Fortran 77*, ed. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Cambridge University Press, 1992.

S2 N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, *J. Chem. Phys.*, 1953, **21**, 1087–1092.

S3 A. D. Koromyslova, A. O. Chugunov and R. G. Efremov, *J. Chem. Inf. Model.*, 2014, **54**, 1189–1199.