Math. Appl. **4** (2015), 109–121
DOI: 10.13164/ma.2015.09

EFFICIENCY COMPARISON OF HOPFIELD NETWORK WITH SIMULATED ANNEALING AS OPTIMIZATION METHODS FOR SOLVING THE TRAVELING SALESMAN PROBLEM

MILOŠ KŘIVAN AND BÁRA BUDÍNSKÁ

Abstract. This paper formulates an open traveling salesman optimization problem and presents a general description of the solution of this problem using Hopfield network and simulated annealing heuristic optimization technique. The experiment described is performed on a set of 45 European cities. The experiment was processed by a specialized computer program.

1. Introduction

With the development of computing technology and the growth of its computational power, there has been an increasing focus on artificial intelligence methods. These methods include terms such as artificial neural networks or evolutionary algorithms. However, their massive utilization in practical applications across all human activities only occurred in the eighties of the previous century, due to the development of personal computers.

Artificial neural networks are used to process and evaluate incomplete, indeterminate or inconsistent information, especially for tasks involving recognition, diagnostics, classification of objects with respect to provided categories or prediction of the time development of a given variable, cluster analysis of multidimensional data, noise filtering and, last but not least, for the solution of special optimization problems, as described in this article.

Evolutionary algorithms are used to find a solution with sufficient quality for large-scale general optimization tasks in a sufficiently short time. Evolutionary algorithms inspired by nature include a whole spectrum of optimization heuristic techniques such as particle swarm or ant colony optimization, genetic algorithms or simulated annealing. Heuristics may be described as a procedure for searching the solution space via shortcuts, which are not guaranteed to find the correct solution but do not suffer from a range of problems of conventional optimization methods such as the requirement of connectivity or differentiability of the criterion or link function, the problem of respecting constraints, being stuck in a shallow local minimum or divergence. However, their application requires the configuration of certain free parameters, which need to be setup based on the specific optimization task – these may, e.g., include the starting or final temperature and the number of

MSC (2010): primary 68T20.
Keywords: traveling salesman problem, continuous Hopfield network, simulated annealing.

iterations of the simulated annealing algorithm described below and based on the evolution of thermodynamic systems. In physics, annealing is a process where an object, heated up to a certain high temperature, is being gradually cooled down to remove internal defects in the object. The high temperature causes the particles in the object to rearrange randomly, which destroys defects in the crystal lattice, and the gradual cooling then allows the particles to stabilize at equilibrium points with a lower probability of new defects being created.

2. Traveling salesman problem

We define a strongly connected directed graph with evaluated edges as an ordered quadruple $[V, E, \varepsilon, w]$:

- V set of vertices,
- E set of edges,
- ε mapping edges with incidence vertices ($\varepsilon : E \to V \times V$),
- w evaluation of edges ($w : \varepsilon(E) \to \mathbb{R}$),

and put $w([i,j]) \equiv w_{ij}$ and $|V| = m$, $n = m^2$, then for open traveling salesman problem [4–11] is following optimization problem:

$$f : \mathbb{R}^n \to \mathbb{R}, \quad f(\vec{x}_0) = \min_{\vec{x} \in \Omega} f(\vec{x}) \ , \quad \Omega \subset \{0,1\}^n, \quad f(\vec{x}) = \sum_i \sum_j w_{ij} x_{ij} \quad (2.1)$$

where x_{ij} is the off/on status of the edge directed from i-th to j-th vertex ($x_{ii} = 0$) for $i, j \in V$.

Each vector $\vec{x} \in \Omega$ selects such a subgraph of the original graph (containing all vertices of original graph) for which it is true that, for every vertex, its indegree and outdegree are equal to one (except the indegree of the unique source vertex and the outdegree of the unique sink vertex).

3. Hopfield network

We define a Hopfield artificial neural network as a directed graph with edges and vertices evaluated dynamically, i.e., as an ordered quintuple $[V, E, \varepsilon, w, y]$:

- V set of vertices (neurons),
- E set of edges (synapses),
- ε mapping edges into incidence vertices ($\varepsilon : E \to V \times V$),
- w evaluation of edges ($w : \varepsilon(E) \to \mathbb{R}$),
- y dynamic evaluation of vertices ($y : V \times \boldsymbol{t} \to \mathbb{R}$)

where $\forall t \in \boldsymbol{t}$ let us put $y([i,t]) \equiv y_i(t)$ and let us put $w([i,j]) \equiv w_{ij}$.

The vector $\vec{w} = [w_{ij} : i, j \in V]$ where $w_{ii} = 0$ is called the *network configuration* and the vector $\vec{y}(t) = [y_i(t) : i \in V]$ is called the *network state* at time t. The state of the network as a vector function of time t is referred to as *active dynamics* of the neural network. Active dynamics of a neural network in continuous time can be defined as a solutions vector of the following system of differential equations [1]:

$$\frac{\mathrm{d}}{\mathrm{d}t} x_j(t) + x_j(t) = \sum_i f_i(x_i(t - \triangle t)) w_{ij} - \vartheta_j \quad (3.1)$$

$i, j \in V$ and then, analogously to biological neural network, we have:

x_i potential of the i-th neuron,
f_i activation function of the i-th neuron ($f_i(x_i) = y_i$),
ϑ_j threshold of the j-th neuron,
w_{ij} synaptic weight links of the i-th neuron to the j-th neuron
Δt signal delay time.

If we replace in (3.1) the derivatives by analogous expressions for discrete time:

$$\frac{\mathrm{d}}{\mathrm{d}t}x_j(t) \equiv \frac{x_j\,(t+1) - x_j\,(t)}{t+1-t}$$

and, if we set $\Delta t = 0$, then we obtain the following system of difference equations and the vector of its solutions defines the active dynamics of a neural network in discrete time:

$$y_j(t+1) = f_j\Big(\sum_i y_i\,(t)\;\; w_{ij} - \vartheta_j\Big), \quad y_i(0) = 0,$$

$i, j \in V$.

Let us approximate the dependence of the state on the potential of the neuron by sigmoid function $f(x) = 1/(1+\mathrm{e}^{-px})$ where the parameter $p > 0$ expresses the slope of the sigmoid. For a slope approaching zero or infinity we get the activation function in the shape of linearity non-linearity, respectively:

$$\lim_{p\to 0} f(x) = \frac{1}{2}, \quad \lim_{p\to\infty} f(x) = 0 \quad \text{for} \quad x < 0, \quad \lim_{p\to\infty} f(x) = 1 \quad \text{for} \quad x > 0.$$

We define the *energy function* of network state [2]

$$E(\vec{y}) = -\Big(\frac{1}{2}\sum_j\sum_i y_i w_{ij} y_j - \sum_j y_j\vartheta_j\Big)$$

and we identify its partial derivatives

$$-\frac{\partial E(\vec{y})}{\partial y_j} = \sum_i y_i w_{ij} - \vartheta_j, \quad -\frac{\partial E(\vec{0})}{\partial y_j} = -\vartheta_j, \quad -\frac{\partial^2 E(\vec{y})}{\partial y_i \partial y_j} = w_{ij},$$

where $i, j \in V$.

We define a *stable state* of the network as a state for which we have $\vec{y}\,(t) = \vec{y}\,(t+1)$ and the parameter T as the inverse value of the sigmoid slope:

Theorem 3.1. *Let us assume $T \to 0$. Then, for all unstable state for $y_i(t) = y_i(t+1)$ where $i \in V - \{j\}$ (asynchronous active dynamics) for $x_j(t+1) \neq 0$ and $w_{ij} = w_{ji}$ for $i, j \in V$, it is true that an energy function value of the network state at time t is greater than the energy function value of the network state at time $t+1$, i.e., $E(\vec{y}(t)) > E(\vec{y}(t+1))$.*

Proof.

$$\Big(\sum_i y_i(t) w_{ij} - \vartheta_j\Big) y_j(t) < \Big(\sum_i y_i(t+1) w_{ij} - \vartheta_j\Big) y_j(t+1),$$

because

$$x_j\,(t+1) > 0 \;\Rightarrow\; y_j\,(t+1) = 1 \;\Rightarrow\; y_j\,(t) = 0,$$
$$x_j\,(t+1) < 0 \;\Rightarrow\; y_j\,(t+1) = 0 \;\Rightarrow\; y_j\,(t) = 1,$$

where $i, j \in V$. □

From the above theorem and from the fact that the energy function is bounded from below, it follows that the network state during the active dynamics by quasi-gradient descent on the energy function converges to a stable state with a locally minimum value of the energy function.

For higher values of the parameter T the convergence to a local minimum of the energy function at the bottom of the smaller gradient slope can be distorted, which can be used for skipping a local minimum. If the value of the parameter T from a sufficiently high initial value will gradually decrease (during active dynamics), then (after skipping a shallow local minima) the network state gets stuck (at a sufficiently low value of the parameter T) in a deep minimum of the energy function at the bottom of the steeper gradient slope and converges to a stable state with approximately the globally minimal value of the energy function, i.e., a network state *freezes* in the global minimum. The parameter T is analogous to the temperature from the simulated annealing optimization method.

Let the number of neurons of Hopfield network equal to m^2 $(m \in \mathbb{N})$, then we can interpret the neurons mentioned as the elements of a square matrix of dimension $m \times m$, and define the following network state function:

$$G_1(\vec{y}) = \frac{1}{2} \sum_i \Big(\sum_j y_{ij} - 1 \Big)^2, \quad G_2(\vec{y}) = \frac{1}{2} \sum_j \Big(\sum_i y_{ij} - 1 \Big)^2,$$

$$G_3(\vec{y}) = \sum_i \sum_j \rho_{ij} \sum_k y_{ik} y_{jk+1},$$

where $i, j \in \{1, \dots, m\}$, $k \in \{1, \dots, m-1\}$ and ρ_{ij} is the metric, and we identify its partial derivatives

$$-\frac{\partial G_1(\vec{0})}{\partial y_{ij}} = 1, \quad -\frac{\partial^2 G_1(\vec{y})}{\partial y_{ij} \partial y_{ik}} = -1,$$

$$-\frac{\partial G_2(\vec{0})}{\partial y_{ij}} = 1, \quad -\frac{\partial^2 G_2(\vec{y})}{\partial y_{ij} \partial y_{kj}} = -1,$$

where $i, j, k \in \{1, \dots, m\}$ and

$$-\frac{\partial G_3(\vec{0})}{\partial y_{ij}} = 0, \quad -\frac{\partial^2 G_3(\vec{y})}{\partial y_{ik} \partial y_{jk+1}} = -\rho_{ij},$$

where $i, j \in \{1, \dots, m\}$ and $k \in \{1, \dots, m-1\}$.

We define the objective function G as the sum of the functions G_1, G_2, G_3 and identify a configuration of the network by extraction from function G using the above partial derivatives, then the objective function G aligns with the energy function E, which will be minimized during active dynamics.

Minimizing the function G_1 or G_2 (see Figure 1) we provide an excitation of just one neuron in each row or column of the above mentioned matrix, i.e., we receive *permissible state* of the network (if we interpret the row or columns of the matrix as the cities or order of its visit), respectively, each permissible network state represents some solution of the traveling salesman problem, but not necessarily optimal.

By minimizing the function G_3 (see Figure 1), we provide the optimal permissible state of the network, i.e., the optimal solution of the traveling salesman

problem, because the value of the function G_3 determines the length of the path of a salesman depending on the permissible state.

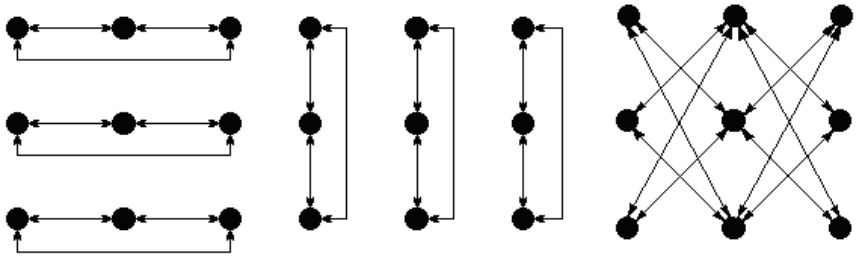

Figure 1. Links providing minimization of functions G_1, G_2, G_3.

4. Simulated annealing

Consider the case that the objective function argument (2.1) unambiguously specifies the macroscopic state of a certain thermodynamic system with energy equal to the function value. Then, we can express its thermodynamic probability

$$P(E_i) = \left|\{\vec{x} \in \mathbb{R}^n : f(\vec{x}) = E_i\}\right|$$

as the number of micro-states corresponding to it.

If we immerse this system in various macro-states with energies E_i in a thermal reservoir, then the Boltzmann equation for the unit size of the Boltzmann constant together with the Taylor expansion of a differentiable function, allows us to express the entropy of the reservoir after the temperatures equilibration for $E = E_0 + E_i = const$ and $E \gg E_i$ as follows:

$$S\left(E_i\right) = S(E) - \frac{\mathrm{d}S(E)}{\mathrm{d}E_i} E_i = \ln P(E - E_i)$$

and then, by using the definition of temperature $\mathrm{d}S(E)/\mathrm{d}E = 1/T$ $(T > 0)$, we can express the thermodynamic probability of a macro-state of the thermal reservoir as a function of the energy of the macro-state of the inserted system, i.e., by the following Boltzmann factor:

$$P(E - E_i) = c\,\mathrm{e}^{-\frac{E_i}{T}}.$$

The simulated annealing algorithm is based on a perturbation of an optimum candidate and a following decision on its replacement by the perturbation in each iteration of the algorithm based on the Metropolis criterion [3]

$$p(\vec{x}_i \to \vec{x}_j) = \frac{P(E_j)}{P(E_i)} = \mathrm{e}^{-\frac{\Delta E}{T}}, \quad \Delta E > 0,$$
$$p(\vec{x}_i \to \vec{x}_j) = 1, \quad \Delta E \leq 0,$$

which expresses the probability of the system transferring from one macro-state to another, where $\Delta E = E_j - E_i$ and $\Delta E/T$ expresses the increase of entropy,

i.e., in accordance with the second law of thermodynamics an impossible event is artificially redefined as a certain event in the specified criterion.

The sequence of accepted perturbations, i.e., acceptable solutions to the optimization task, forms a Markov chain with memory of order one, i.e., the occurrence of the given solution is only conditioned by the occurrence of the previous solution. The perturbations which lie outside the area of admissible solutions are automatically rejected.

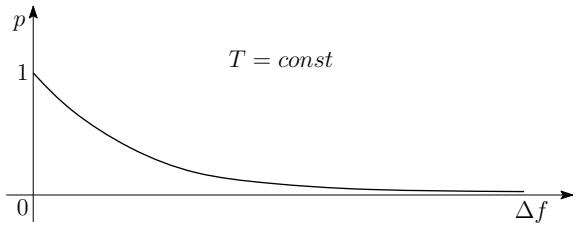

Figure 2. Dependence of probability on increase of energy.

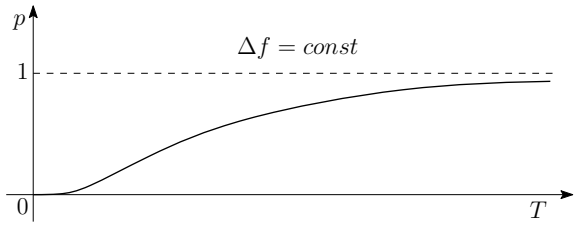

Figure 3. Dependence of probability on temperature.

From $p(\Delta f)$ (see Figure 2) it is clear that a significantly "worse" solution is accepted with respect to the previous solution at a much lower probability than a slightly "worse" solution. $p(T)$ (see Figre 3) may be used to control the probability of the acceptance of the solution during the iteration cycle. We initiate the iteration cycle with a sufficiently high temperature to ensure that almost every proposed solution is accepted for a certain period of time, which will allow an initial approximation of the solution to "escape" areas with shallow local minima. Later on, we reduce the temperature so that almost no "worse" solution is accepted, i.e., during the iteration cycle, we cool down the system representing the optimization task from a sufficiently high temperature to a sufficiently low temperature until a solution is "frozen" in a sufficiently deep local minimum (see Figure 4). The temperature descent may be modeled, e.g., as an exponentially decreasing function

$$T = T_0 \mathrm{e}^{-\frac{iter}{\tau}}, \quad \tau = -\frac{N}{\ln(T_\infty/T_0)}, \quad T_\infty \approx \lim_{iter\to\infty} T_0 \mathrm{e}^{-\frac{iter}{\tau}} = 0$$

where T_0 or T_∞ are the initial or final temperatures, respectively, and N is the number of iterations of the algorithm.

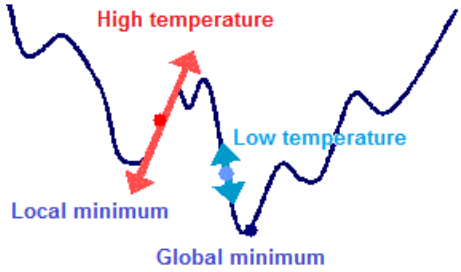

Figure 4. Freezing of solution.

5. Experiment

The following computational experiment was performed on a set of 45 European cities (see Table 1) using both optimization methods mentioned for two ways of temperature descent, i.e., exponential and linear. The experiment was always performed for variant values of initial temperature (0.1÷1000) or values of parameter SEED (0÷1000) and for a variant number of iterations ($10^3 \div 10^6$) or ($10^3 \div 10^9$), see Table 2 (Hopfield network) and Table 3 (simulated annealing) including the average optimal distance $\varnothing$ and the absolute deviation σ. Table 3 implies Figure 5. All simulated annealing experiments were performed for the initial temperature equal to one, except for the last series (10^9 iterations) of experiments for linear descent where the initial temperature was set to five.

A fragment of Fortran source code implementing asynchronous active dynamics of Hopfield network follows:

```
      .
      .
C
C     ACTIVATION FUNCTION DEFINITION
C
      F(X) = 1.0/(1.0 + EXP(-X/Temperature))
C
C     A C T I V E    D Y N A M I C S    (ASYNCHRONOUS)
C
C     N - NUMBER OF CITIES
C     Y - STATE OF NEURON
C     W - SYNAPTIC WEIGHT
C
      DO I=1,N*N
      Y(I) = 1.0
      DO J=1,N*N
      Y(I) = Y(I) + Y(J)*W(J+(I-1)*N)
      ENDDO
      Y(I) = F(Y(I))
      ENDDO
      .
      .
```

A fragment of Fortran source code implementing variant ways of temperature descent follows:

```
      FUNCTION TSP (T0,Tinf,LIM,SEED)
C
C     T0   - INITIAL TEMPERATURE
C     Tinf - FINAL TEMPERATURE
C     LIM  - LIMIT OF ITERATION
C     SEED - BEGINNING OF PSEUDORANDOM SERIES
C
      ITERATION = -1
      TAU = -LIM/LOG(Tinf/T0)
100   ITERATION = ITERATION + 1
      IF(DESCENT.EQ.0)THEN                  ! 0-EXPONENTIAL, 1-LINEAR
      Temperature = T0*EXP(-ITERATION/TAU)
      ELSE
      Temperature = -((T0-Tinf)*ITERATION/LIM) + T0
      ENDIF
      .
      .
      .
      IF(ITERATION.LT.LIM)GOTO 100
      RETURN
```

Amsterdam	Dublin	Kobenhavn	Munchen	Strasbourg
Ankara	Dubrovnik	Lisboa	Narvik	Venezia
Athenai	Edinburgh	Liverpool	Oslo	Warszawa
Barcelona	Frankfurt	London	Palermo	Wien
Beograd	Geneve	Luxenbourg	Paris	Zurich
Berlin	Hamburg	Madrid	Praha	
Bratislava	Hammerfest	Malaga	Roma	
Bruxelles	Helsinki	Marseille	Salzburg	
Bucuresti	Instanbul	Milano	Sofia	
Budapest	Kijev	Moskva	Stockholm	

Table 1. Used European cities.

Exp	0.1	1	5	10	50	100	500	1000	Ø	σ
3	63599	44225	43778	42580	47373	43111	43770	46660	46887	4300
4	63599	35968	35403	36682	37668	40212	36563	41637	40967	5826
5	34278	38222	38557	34278	33610	37330	35180	39517	36372	2035
6	38912	35892	35559	33668	35559	38912	35892	35559	36244	1334

Lin	0.1	1	5	10	50	100	500	1000	Ø	σ
3	63599	49691	56409	63494	ERR	ERR	ERR	ERR	58298	5248
4	63599	35818	39703	49691	56409	63599	ERR	ERR	51470	9733
5	78595	37522	40870	35818	39703	45432	57134	63317	49799	12412
6	78595	35559	35559	35815	36239	35818	40657	40764	42376	9055

Table 2. Optimal distance for variant initial temperature and number of iterations.

Exp	0	1	5	10	50	100	500	1000	Ø	σ
3	39544	39009	39315	40824	38577	42079	42890	36130	39796	1601
4	30663	33507	26551	29417	33686	30116	31876	31343	30895	1708
5	27146	26944	27969	30623	29658	27899	27421	28596	28282	1008
6	26534	25932	29001	27070	26157	28419	27805	26466	27173	927
7	26528	25460	26452	25361	24428	26454	29066	27468	26402	989
8	25934	24786	24557	26812	25545	26471	28403	25428	25992	928
9	33664	33617	31528	31320	26114	29320	27255	30521	30417	2141

Lin	0	1	5	10	50	100	500	1000	Ø	σ
3	48024	54057	48038	49988	53922	53167	42890	51830	50240	3005
4	40581	38116	36496	37282	35368	40152	37476	36299	37721	1421
5	29916	29618	29475	31216	29126	29384	28417	30391	29693	611
6	27273	27171	25851	28421	26072	26830	27566	27584	27096	634
7	26904	25917	25618	25798	25442	26152	26941	26758	26191	507
8	25191	24922	25343	26162	24741	26373	24504	25713	25369	536
9	24360	24074	24881	25577	25071	24558	24924	25811	24907	439

Table 3. Optimal distance for variant initial seed and number of iterations.

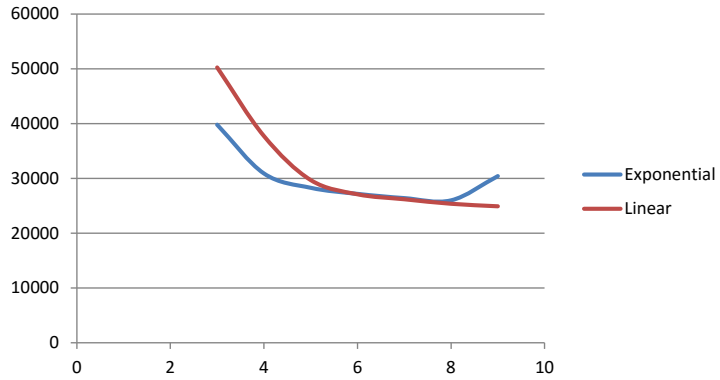

Figure 5. Average optimal distance dependent on number of iterations.

On average, the best results we obtain when the number of iterations is equal to 10^6 for Hopfield network (see Table 2) and 10^8 or 10^9 for simulated annealing (see Table 3) at exponential or linear descent, respectively.

The best optimal solution of Hopfield network (see Table 4 and Figure 6) or simulated annealing (see Table 5 and Figure 7) is 33 610 km or 24 074 km, respectively:

Hopfield Network:

```
Initial Temp   Final Temp   Iteration Limit    Distance0  Distance1  Saving
      50.000     0.000001            100000       93698.     33610.  60088.
```

Simulated Annealing:

```
Initial Temp  Final Temp  Iteration Limit SEED Distance0  Distance1  Saving
       5.000    0.000001       1000000000    1    77784.     24074.  53710.
```

where `Distance0` or `Distance1` is a randomly generated initial solution or optimal solution, respectively.

Lisboa	Edinburgh	Zurich	Instanbul	Oslo
Madrid	Liverpool	Salzburg	Sofia	Stockholm
Barcelona	London	Munchen	Beograd	Helsinki
Malaga	Strasbourg	Venezia	Budapest	Narvik
Marseille	Warszawa	Milano	Wien	Hammerfest
Geneve	Bucuresti	Roma	Bratislava	
Paris	Kijev	Dubrovnik	Praha	
Amsterdam	Frankfurt	Palermo	Berlin	
Bruxelles	Luxenbourg	Athenai	Hamburg	
Dublin	Moskva	Ankara	Kobenhavn	

Table 4. Hopfield network solution.

Lisboa	Athenai	Salzburg	London	Helsinki
Malaga	Ankara	Munchen	Dublin	Stockholm
Madrid	Instanbul	Zurich	Liverpool	Oslo
Barcelona	Bucuresti	Geneve	Edinburgh	Narvik
Marseille	Sofia	Strasbourg	Kobenhavn	Hammerfest
Milano	Beograd	Frankfurt	Hamburg	
Venezia	Budapest	Luxenbourg	Berlin	
Roma	Bratislava	Paris	Warszawa	
Dubrovnik	Wien	Bruxelles	Kijev	
Palermo	Praha	Amsterdam	Moskva	

Table 5. Simulated annealing solution.

The calculation experiment was realized on the usual laptop CPU Intel 2GHz and the calculation time took about five minutes for simulated annealing (10^9 iterations) and one hour for Hopfield network (10^6 iterations) depending on the network state stabilization moment. The simulated annealing algorithm is thus much faster than the Hopfield network quasi-gradient descent. The source vertex is equal to Lisbon and the sink vertex is equal to Hammerfest in both cases. The simulated annealing can be used for an asymmetric travelling salesman problem but the Hopfield network can only be used only for a symmetric travelling salesman problem.

6. Conclusion

The exact optimal solution is 23 932 km (computed by IBM ILOG CPLEX Optimizer), whereas the best optimal solution computed by simulated annealing is 24 074 km, which is only by about 142 km (0.6%) worse.

The simulated annealing algorithm yields much better results than the Hopfield network quasi-gradient descent, i.e., about 10 000 km shorter distance. The trajectory of the Hopfield network quasi-gradient descent apparently never crossed the area of a sufficiently deep local minimum during our experiment. The choice of the weighting coefficient w for equalizing the numerical disbalance between functions $G_1 + G_2$ and G_3 ($G = G_1 + G_2 + wG_3$) can be substituted by setting a threshold value of the neurons. In the above experiment, threshold value of each neuron was set identically to an initial value because the metric ρ was normalized (see part Hopfield network).

Figure 6. Hopfield network solution.

The initial temperature of the simulated annealing is set so that, during the first 10%, a sufficient number of iterations have been accepted by almost every perturbation. The rate of temperature decrease is inversely proportional to the selected number of iterations varied in the above experiment. The experiment implies that the exponential descent (as compared with the linear descent) of a temperature is preferable for smaller number of iteration.

The source and sink vertex were determined by the program used in the experiment spontaneously and identically by both optimization methods used above. The description of the computational experiment implies a high efficiency of the

Figure 7. Simulated annealing solution.

optimization algorithm of simulated annealing when searching the admissible solution space, given by the ratio of the total number of admissible solutions (45!) to the number of simulated admissible solutions, i.e., about $10^{56} : 10^{9}$. The simulated annealing algorithm can be run in parallel (for example) in eight threads on a quad-core computer for eight variously selected values of the initial temperature and then the best obtained solution can be selected.

References

[1] S. Grossberg, *The Adaptive Brain*, North-Holland, 1987.
[2] J. J. Hopfield, *Artificial neural networks*, IEEE Circuits Device. **4** (1988), 3–10.
[3] N. Metropolis, *Equations of state calculations by fast computing machines*, J. Chem. Phys. **21** (1953), 1087–1092.
[4] M. Fischetti and P. Toth, *An additive bounding procedure for the asymmetric travelling salesman problem*, Math. Program. **53** (1992), 173–197.
[5] D. Fogel, *Applying evolutionary programming to selected traveling salesman problems*, Cybernet. Syst. **24** (1993), 27–36.
[6] M. Junger, G. Reinelt and G. Rinaldi, *The traveling salesman problem*, Elsevier Science, 1995.
[7] G. Laporte, *The vehicle routing problem: an overview of exact and approximate algorithms*, Oper. Res. *59* (1992), 345–358.
[8] S. Lin S. and B. W. Kernighan, *An effective heuristic algorithm for the traveling salesman problem*, Oper. Res. **21** (1973), 498–516.
[9] J. Y. Potvin, *The traveling salesman problem: A neural network perspective*, ORSA J. Comput. **5** (1993), 328–347.

[10] W. Z. Rao, C. Jin and Y. Y. Huang, *Hybrid algorithm of the nearest neighbor and insertion for the traveling salesman problem*, Systems Engineering – Theory & Practice **31** (2011), 1419–1428.

[11] F. G. Zhao, J. S. Sun and S. J. Li, *A hybrid genetic algorithm for the traveling salesman problem with pickup and delivery*, IJAC **6** (2009), 97–102.

Miloš Křivan, Faculty of Informatics and Statistics of University of Economics in Prague
e-mail: krivan@intelligentsoftware.eu

Bára Budínská, Department of Economics and Management of Czech Technical University in Prague