# A New Method of DDB Logical Structure Synthesis Using Distributed Tabu Search

Eduard Babkin[1] and Margarita Karpunina[2],

National Research University "Higher School of Economics"
Dept. of Information Systems and Technologies,
Bol. Pecherskaya, 25,
6030155 Nizhny Novgorod, Russia
[1] eababkin@hse.ru
[2] karpunina-margarita@yandex.ru

**Abstract.** In this paper we propose a parallel tabu search algorithm based on the consecutive tabu algorithm constructed by us earlier to solve the problem of the distributed database optimal logical structure synthesis. Also we provide a reader with some information about the performance of our new parallel algorithm and the quality of the solutions obtained with help of it.

**Keywords:** Neural networks, tabu search, genetic algorithms, parallel programming, distributed databases.

## 1 Introduction

The problems of decomposition of complex data structures play an extremely important role in many critical applications varying from cloud computing to distributed databases (DDB) [7]. In later class of applications that problem is usually formulated as synthesis of optimal logical structure (OLS). In accordance with [3] it consists of two stages. The first stage is decomposition of data elements (DE) into logical record (LR) types. The second stage is irredundant replacement of LR types in the computing network. For each stage various domain-specific constraints are introduced (like irredundant allocation, semantic contiguity of data elements, available external storage) as well as optimum criteria are specified. In our work the criterion function is specified as a minimum of total time needed for consecutive processing of a set of DDB users' queries [8].

From mathematical point of view the specified problem is a NP-complete non-linear optimization problem of integer programming. So far different task-specific approaches were proposed such as branch-and-bound method with a set of heuristics (BBM) [8], probabilistic algorithms, etc. However not many of them exploit benefits of parallel processing and grid technologies [4], [5], [6], [9].

In previous works the authors developed the exact mathematical formalization of the OLS problem and offered the sequential tabu search (TS) algorithm which used different Tabu Machines (TMs) for each stage of the solution [1], [2]. The constructed algorithm produced solutions with good quality, but it was computationally efficient for small mock-up problems.

In the present article we propose a new distributed model of TM (DTM) and a computationally efficient parallel algorithm for solutions of complex data structure decomposition problems. The article has the following structure. In Section 2 we outline critical elements of TM. In Section 3 general description of DTM algorithm is given and Section 4 specifies it in details. Section 5 briefly describes evaluation of the obtained parallel algorithm. Overview of the results in Section 6 concludes the article.

## 2 Short Overview of Tabu Machine Model and Dynamics

In our work we use the generic model of TM as it was specified by Minghe Sun and Hamid R. Nemati [10] with the following important constituents.

$S = \{s_1, ..., s_n\}$ is the current state of the TM, it is collectively determined by the states of its nodes.

$S_0 = \{s_1^0, ..., s_n^0\}$ is the state of the TM with the minimum energy among all states which are obtained by the current moment within the local period (or within the short term memory process (STMP)).

$S_{00} = \{s_1^{00}, ..., s_n^{00}\}$ is the state of the TM with the minimum energy among all states which are obtained by the current moment (within both the STMP and the long term memory process (LTMP)).

$T = \{t_1, ..., t_n\}$ is a vector to check the tabu condition.

$E(S)$ is the TM energy corresponding to the state $S$.

$E(S_0)$ is the TM energy corresponding to the state $S_0$.

$E(S_{00})$ is the TM energy corresponding to the state $S_0$.

$k$ is the number of iterations (i.e. the number of neural network (NN) transitions from the one state to another) from the outset of the TM functioning.

$h$ is the number of iterations from the last renewal the value of $E(S_0)$ within the STMP.

$c$ is the number of the LTMPs carried out by the current moment.

The following variables stand as parameters of the TM-algorithm:

$l$ – the tabu size,

$\beta$ – the parameter determining the termination criterion of the STMP,

$C$ – maximum number of the available LTMPs inside the TM-algorithm.

The state transition mechanism of the TM is governed by TS and performed until the predefined stopping rule is satisfied. Let's name this sequence of state transitions as a work period of the TM. It is advisable to run the TM for several work periods. It is better to begin a new work period of the TM using information taken from the previous work periods, from a "history" of the TM work by applying a LTMP. In such a case a TS algorithm finds a node which has not changed its state for the longest time among all neurons of the TM. And then this node is forced to switch its state.

## 3 A Consecutive TM-Algorithm for OLS Problem

As [3] states, the general problem of DDB OLS synthesis consists of two stages.

1. **Composition of logical record (LR) types** from data elements (DE) using the constraints on: the number of elements in the LR type; single elements inclusion in the LR type; the required level of information safety of the system. In addition, LR types synthesis should take into account semantic contiguity of DE.
2. **Irredundant allocation of LR types** among the nodes in the computing network using the constraints on: irredundant allocation of LR types; the length of the formed LR type on each host; the total number of the synthesized LR types placed on each host; the volume of accessible external memory of the hosts for storage of local databases; the total processing time of operational queries on the hosts.

The synthesis objective is to minimize the total time needed for consecutive processing of a set of DDB users' queries. Such problem has an exact but a very large mathematical formalization. So, we provide it in the Appendix I and Appendix II of this paper due to its limited size and should refer to [1], [2], [3], [8] for further details.

In our previous work [2] we have offered a new method for formalization of the described problem in the terms of TM and have constructed TMs' energy functions as follows. TM for the first stage consists of one layer of neurons, connected by complete bidirectional links. The number of neurons in the layer is equal to $I^2$, where $I$ is the number of DEs. Each neuron is supplied with two indexes corresponding to numbers of DEs and LRs. For example, $OUT_{xi} = 1$ means, that the DE $x$ will be included to the $i$-th LR. All outputs $OUT_{xi}$ of a network have a binary nature, i.e. accept values from set $\{0,1\}$. The following TM energy function for LR composition was proposed:

$$E = -\frac{1}{2} \cdot \sum_{i=1}^{I} \sum_{j=1}^{I} \sum_{x=1}^{I} \sum_{y=1}^{I} \left[ -A_1 \cdot \delta_{xy} \cdot \left(1 - \delta_{ij}\right) + B_1 \cdot \delta_{ij} \cdot \left(1 - \delta_{xy}\right) \cdot \left(2 \cdot a_{xy}^{g} - 1\right) - D_1 \cdot \delta_{ij} \cdot \right.$$
$$\left. \cdot \left(incomp\_gr_{xy} + incomp\_gr_{yx}\right) \right] \cdot OUT_{xi} \cdot OUT_{yj} + \sum_{i=1}^{I} \sum_{x=1}^{I} \left[ \frac{B_1}{2} \cdot \sum_{\substack{y=1 \\ y \neq x}}^{I} \left(a_{xy}^{g}\right)^2 + \frac{C_1}{2 \cdot F_i} \right] \cdot OUT_{xi} \qquad \mathbf{(1)}$$

Here $w_{xi,yj} = -A_1 \cdot \delta_{xy} \cdot \left(1 - \delta_{ij}\right) + B_1 \cdot \delta_{ij} \cdot \left(1 - \delta_{xy}\right) \cdot \left(2 \cdot a_{xy}^{g} - 1\right) - D_1 \cdot \delta_{ij} \cdot \left(incomp\_gr_{xy} + \right.$ $\left. + incomp\_gr_{yx}\right)$ are weights of neurons, $T_{xi} = \left( \frac{B_1}{2} \cdot \sum_{\substack{y=1 \\ y \neq x}}^{I} \left(a_{xy}^{g}\right)^2 + \frac{C_1}{2 \cdot F_i} \right)$ are the neurons' thresholds.

For the second stage of irredundant LR allocation we offered TM with the same structure as TM for LR composition, but the number of neurons in the layer is equal to $T \cdot R_0$, where $T$ is the number of LRs, synthesized during LR composition, $R_0$ is the number of the hosts available for LR allocation.

As a result of constraints translation into the terms of TM the following TM energy function for the LR allocation was obtained:

$$E = -\frac{1}{2} \cdot \sum_{r_1=1}^{R_0} \sum_{r_2=1}^{R_0} \sum_{t_1=1}^{T} \sum_{t_2=1}^{T} \left[ -A_2 \cdot \delta_{t_1 t_2} \cdot \left(1-\delta_{r_1 r_2}\right) \right] \cdot OUT_{t_1 r_1} \cdot OUT_{t_2 r_2} + \sum_{r_1=1}^{R_0} \sum_{t_1=1}^{T} \left[ \frac{B_2 \cdot \psi_0}{2 \cdot \theta_{t_1 r_1}} \cdot \sum_{i=1}^{I} \left( x_{it_1} \cdot \rho_i \right) + \frac{C_2}{2 \cdot h_{r_1}} + \frac{D_2 \cdot \psi_0}{2 \cdot \eta_{r_1}^{EMD}} \cdot \sum_{i=1}^{I} \left( \rho_i \cdot \pi_i \cdot x_{it_1} \right) + \frac{E_2 \cdot \left( t_{r_1}^{srh} + t_{r_1} \right)}{2} \cdot \sum_{p=1}^{P_0} \left( \frac{SN_{pt_1}}{T_p} \right) \right] \cdot OUT_{t_1 r_1} \qquad \mathbf{(2)}$$

Here $w_{t_1 r_1, t_2 r_2} = -A_2 \cdot \delta_{t_1 t_2} \cdot \left(1-\delta_{r_1 r_2}\right)$ are weights of neurons,

$$T_{t_1 r_1} = \frac{B_2 \cdot \psi_0}{2 \cdot \theta_{t_1 r_1}} \cdot \sum_{i=1}^{I} \left( x_{it_1} \cdot \rho_i \right) + \frac{C_2}{2 \cdot h_{r_1}} + \frac{D_2 \cdot \psi_0}{2 \cdot \eta_{r_1}^{EMD}} \cdot \sum_{i=1}^{I} \left( \rho_i \cdot \pi_i \cdot x_{it_1} \right) + \frac{E_2 \cdot \left( t_{r_1}^{srh} + t_{r_1} \right)}{2} \cdot \sum_{p=1}^{P_0} \left( \frac{SN_{pt_1}}{T_p} \right)$$

are the neurons' thresholds. Here the $I$ is the number of DEs, $z_{pr_1}^{t_1} = OUT_{t_1 r_1} \cdot SN_{pt_1}$ and $SN_{pt_1}$ is introduced as a normalized sum, i.e. $SN_{pt_1} = \begin{cases} 1, \text{ if } \sum_{i=1}^{I} w_{pi}^{Q} x_{it_1} \geq 1 \\ 0, \text{ if } \sum_{i=1}^{I} w_{pi}^{Q} x_{it_1} = 0 \end{cases}$, where $w_{pi}^{Q}$ is the matrix of dimension $\left(P_0 \times I\right)$, that matrix shows which DEs are used during processing of different queries.

In [2] we also compared the developed TM-algorithm with other methods like [10] to estimate an opportunities and advantages of TS over our earlier approaches based on Hopfield Networks or their combination with genetic algorithms (NN-GA-algorithm) [3]. For complex mock-up problems we obtained the TM solutions with the quality higher than the quality of solutions received with help of NN-GA-algorithm on average 8,7%, the quality of solutions received with help of BBM on average 23,6% (refer to Fig. 1), and CPU time for LR composition was on average 36% less that the same spent by the Hopfield Network approach. So, our TM is able to produce good solutions. But nevertheless this algorithm is time consuming on high-dimensional tasks, and therefore it is needed to construct a parallel TM-algorithm in order to validate our approach on the high-dimensional tasks and increase the performance. Moreover, the parallel algorithm helps us to reveal the influence of the tabu parameters on the tasks' solution process and to determine the dependency between the tabu parameters and characteristics of our problem in order to obtain the better solutions faster.

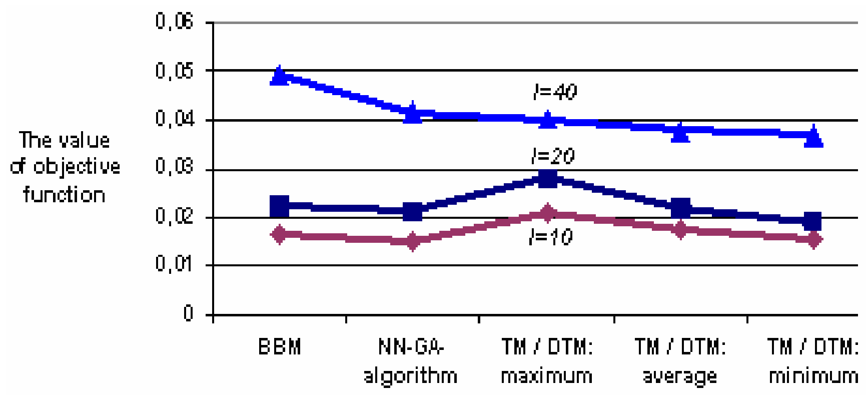

**Fig. 1.** The value of objective function on mock-up problems solutions.

## 4 A General Description of DTM Functioning

The proposed parallel algorithm of TM exploits parallelization capabilities of the following procedures: finding a neuron to change its state; changing the value of $\Delta E(S_i)$ of neurons for using it on the next iteration; the calculation of energy function value; the calculation of values of auxiliary functions used in aspiration criteria of TM; the transition from one local cycle to the other.

For the case of the homogeneous computational parallel cluster with multiple identical nodes the following general scheme of new parallel functionality is proposed. The set of neurons of the whole TM is distributed among all nodes' processors according to the formula $N_p = \begin{cases} n_1 + 1, \text{ if } p < n_2 \\ n_1, \text{ otherwise} \end{cases}$, where $n_1 = \left\lfloor \frac{N}{P} \right\rfloor, n_2 = N \bmod P$, $N$ – the number of neurons in the whole TM, $p = \overline{0,(P-1)}$ – the index of processor, $P$ – the number of processors. The number of Tabu sub-machines (TsMs) is equal to the number of available processors. So, one TsM is located on each processor and TsM with index $p$ consists of $N_p$ neurons. During the initialization stage neural characteristics are set to each neuron. The scheme of DTM is depicted on Fig. 2. The same figure shows how the weight matrix of each TsM $W_p = \left\{ w_{ij}^p; i = \overline{1,N_p}; j = \overline{1,N} \right\} =$ $= \left\{ w_{ij}; i = \overline{\sum_{k=0}^{p-1} N_k + 1, \sum_{k=0}^{p} N_k}; j = \overline{1,N} \right\}$ is constructed from the weight matrix $W = \left\{ w_{ij}; i,j = \overline{1,N} \right\}$ of the whole TM. When the optimal state of DTM is achieved, the results from all TsMs are united. The proposition that the energy of the whole TM is additive on the energies of TsMs including in the DTM, i.e. $E = E_0 + E_1 + \ldots + E_{P-1} = \sum_{p=0}^{P-1} E_p$, is formulated and proofed by the authors but due to lack of the space is omitted in that article.

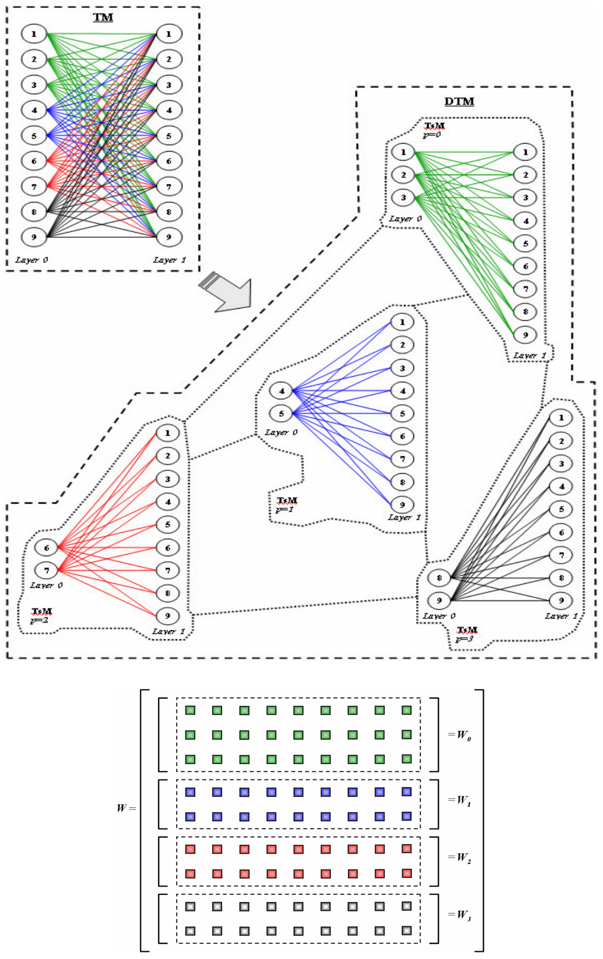

**Fig. 2.** DTM scheme (*up*) and method of $W_p$ construction from the whole matrix $W$ (*bottom*).

Let's consider a common implementation of DTM taking into account a parallel implementation of foregoing procedures.

**Initialization.** At this stage we assume that TsMs included into DTM are constructed and initialized. Construction and initialization are conducted following the mentioned above scheme of distribution of DTM neurons among the set of available processors. After the structure of each TsM is defined, TsMs are provided with the following characteristics: the matrix of neurons weights, vector of neurons thresholds, and vector of neurons biases. Thus, on the current stage we have the set of TsMs, and the elements of this set are

$$subTM_p = \{W_p, I_p, T_p, In_p\}, \quad p = \overline{0,(P-1)} \ , \tag{3}$$

where $subTM_p$ – $p$-th TsM, $W_p$ – the matrix of its neurons weights, $I_p$ – the vector of neurons biases, $T_p$ – the vector of neurons thresholds, and $In_p$ – the vector of initial states of TsM's neurons. Matrixes $W_p$ and vectors $I_p$ and $T_p$ are defined according to the following formulas:

$$W = \{w_{ij}; i, j = \overline{1,N}\} = \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{P-1} \end{bmatrix} = \begin{bmatrix} \{w_{ij}^0; i = \overline{1,N_0}; j = \overline{1,N}\} \\ \{w_{ij}^1; i = \overline{1,N_1}; j = \overline{1,N}\} \\ \vdots \\ \{w_{ij}^{P-1}; i = \overline{1,N_{P-1}}; j = \overline{1,N}\} \end{bmatrix} = \begin{bmatrix} \{w_{ij}; i = \overline{1,N_0}; j = \overline{1,N}\} \\ \{w_{ij}; i = \overline{N_0+1, N_0+N_1}; j = \overline{1,N}\} \\ \vdots \\ \left\{w_{ij}; i = \overline{\sum_{k=0}^{P-2} N_k + 1, \sum_{k=0}^{P-1} N_k}; j = \overline{1,N}\right\} \end{bmatrix} \tag{4}$$

$$I = \{i_j; j = \overline{1,N}\} = \begin{bmatrix} I_0 \\ I_1 \\ \vdots \\ I_{P-1} \end{bmatrix} = \begin{bmatrix} \{i_j^0; j = \overline{1,N_0}\} \\ \{i_j^1; j = \overline{1,N_1}\} \\ \vdots \\ \{i_j^{P-1}; j = \overline{1,N_{P-1}}\} \end{bmatrix} = \begin{bmatrix} \{i_j; j = \overline{1,N_0}\} \\ \{i_j; j = \overline{N_0+1, N_0+N_1}\} \\ \vdots \\ \left\{i_j; j = \overline{\sum_{k=0}^{P-2} N_k + 1, \sum_{k=0}^{P-1} N_k}\right\} \end{bmatrix} \tag{5}$$

$$T = \{t_j; j = \overline{1,N}\} = \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_{P-1} \end{bmatrix} = \begin{bmatrix} \{t_j^0; j = \overline{1,N_0}\} \\ \{t_j^1; j = \overline{1,N_1}\} \\ \vdots \\ \{t_j^{P-1}; j = \overline{1,N_{P-1}}\} \end{bmatrix} = \begin{bmatrix} \{t_j; j = \overline{1,N_0}\} \\ \{t_j; j = \overline{N_0+1, N_0+N_1}\} \\ \vdots \\ \left\{t_j; j = \overline{\sum_{k=0}^{P-2} N_k + 1, \sum_{k=0}^{P-1} N_k}\right\} \end{bmatrix} \tag{6}$$

Vector $In$ of initial states of the whole TM neurons is random generated, and then cut on $P$ parts, each of which (i.e. $In_p$) is corresponded to the concrete TsM.

**The local cycle of the TM.** Let's consider the local cycle of DTM.

*Choose the neuron-candidate for the next move.* At the first step of the TM local cycle we search for neuron on each TsM, which should change its state on current iteration. The criterion to choose such a neuron is defined as the following:

$$\Delta E_p(S_j) = \left\{\min\left\{\Delta E_p(S_i) \mid i = \overline{1,N_p}\right\} \ : \ k - t_j \le l \ \vee \ E_p(S) + \Delta E_p(S_j) < E_p(S_0)\right\}. \tag{7}$$
$$p = \overline{0,(P-1)}$$

Thus, the search of neurons satisfied to the condition (7) is performed in parallel on the hosts of CN.

*The comparison of found neurons.* After the neuron satisfied to the condition (7) is found on each host, the search with help of STMP reduce operations defined by authors for MPI_Allreduce function is performed within the whole DTM to find the neuron $j^*$, such that $\Delta E(S_{j^*}) = \min\left\{\Delta E_p(S_j) \mid p = \overline{0,(P-1)}\right\}$.

*Change the energy value of neurons.* After the required neuron $j^*$ has been found, and each TsM has information about it, each neuron of $subTM_p$, $p = \overline{0,(P-1)}$ changes its $\Delta E(S_i)$ value. The calculation of DTM energy function change is done in parallel on each $subTM_p$. Further the cycle is repeated following described scheme until the condition of exit from the local cycle of the TM is satisfied.

**The global cycle of the TM.** We select neuron, that didn't change its state longest, on each TsMs. The number $j$ of this neuron on each $subTM_p$ is defined according to the following criteria:

$$(t_j)_p = \min\left\{t_i \mid i = \overline{1,N_p}\right\}, \quad p = \overline{0,(P-1)}. \tag{8}$$

The search of $(t_j)_p$ is done on the available processors in parallel according to the formula (8).

*The comparison of found neurons.* After the neuron satisfied to the condition (8) is found on each host, the search with help of LTMP reduce operations defined by authors for MPI_Allreduce function is performed within the whole DTM to find the neuron $j^*$, such that $t_{j^*} = \min\left\{(t_j)_p \mid p = \overline{0,(P-1)}\right\}$.

*Change the energy value of neurons.* After the required neuron $j^*$ has been found, and each TsM has information about it, each neuron of $subTM_p$, $p = \overline{0,(P-1)}$ changes its $\Delta E(S_i)$ value. The calculation of DTM energy function change is done in parallel on each $subTM_p$. Further the cycle is repeated following described scheme until the number of LTMP calls will exceed $C : C \in Z^+, C \geq 0$ times. After that the search is stopped and the best found state is taken as the final DTM state.

## 5 The Algorithm of DTM Functioning

Let's try to represent the general description as an algorithm outlined step by step. We will use the following notations: $N$ – the number of neurons in the DTM, i.e. $|S| = |S_0| = |S_{00}| = N$; $N_p$ – the number of neurons including into the TsM $subTM_p$, where $p = \overline{0,(P-1)}$; $P$ – the number of processors on which DTM operates.

**Step 1.** Construct TsMs $subTM_p$ and randomly initialize initial states of its neurons. Define the tabu-size $l$ of DTM. Let $h=0$, $k=0$ – counters of iterations in the frame of the whole DTM. Let $c=0$ and $C\geq 0$ – the maximum number of LTMP calls in the frames of the whole DTM. Let $\beta>0$ is defined according to inequality $\beta\cdot N>l$ in the frames of the whole DTM too.

**Step 2.** Find the local minimum energy state $S_0$. Calculate $E(S_0)$ and

$$\Delta E(S)=\begin{bmatrix}\Delta E(S_1)\\ \Delta E(S_2)\\ \vdots\\ \Delta E(S_N)\end{bmatrix}=\begin{bmatrix}\Delta E_0(S_i),\ i=\overline{1,N_0}\\ \Delta E_1(S_i),\ i=\overline{N_0+1,N_0+N_1}\\ \vdots\\ \Delta E_{P-1}(S_i),\ i=\overline{\sum_{k=0}^{P-2}N_k+1,\sum_{k=0}^{P-1}N_k}\end{bmatrix},\quad i=\overline{1,N}. \qquad \mathbf{(9)}$$

The values of $E_p(S_0)$ and $\Delta E_p(S_i)$ for $p=\overline{0,(P-1)}$ are calculated in parallel on $P$ processors. Let $S_{00}=S_0$ is the best global state, and $E(S_{00})=E(S_0)$ is the global minimum of energy. Let $S=S_0$ and $E(S)=E(S_0)$. Let $t_i=-\infty,\forall i=\overline{1,N}$.

**Step 3.** In the frames of each $subTM_p$ choose the neuron $j$ with $\Delta E_p(S_j)$ satisfied to $\Delta E_p(S_j)=\left\{\min\left\{\Delta E_p(S_i)\,|\,i=\overline{1,N_p}\right\}\ :\ k-t_j\leq l\ \vee\ E_p(S)+\Delta E_p(S_j)<E_p(S_0)\right\}, p=\overline{0,(P-1)}$.

**Step 4.** Using STMP reduce operations defined by authors, form the set $\left\{j^*,\ \Delta E(S_{j^*}),\ \ s_{j^*}\right\}$, where $j^*$ – the index of neuron (in the frames of the whole DTM) changing its state at the current moment, $\Delta E(S_{j^*})$ – the change of DTM energy function value after the neuron $j^*$ has changed its state, $s_{j^*}$ – the new state of neuron $j^*$.

**Step 5.** If $subTM_p$ contains the neuron $j^*$, then $t_{j^*}=k$, $s_{j^*}=1-s_{j^*}$.

**Step 6.** Let $t_{j^*}=k$, $k=k+1$, $h=h+1$, $S=S_{j^*}$, $E(S)=E(S)+\Delta E(S_{j^*})$ in the frames of the whole DTM.

**Step 7.** Update $\Delta E(S)$ using (9). The values of $\Delta E_p(S_i)$ are calculated in parallel on $P$ processors.

**Step 8.** Determine if the new state $S$ is the new local and / or global minimum energy state: if $E(S)<E(S_0)$, then $S_0=S$, $E(S_0)=E(S)$ and $h=0$; if $E(S)<E(S_{00})$, then $S_{00}=S$ and $E(S_{00})=E(S)$ in the frames of the whole DTM.

**Step 9.** If $h<\beta\cdot N$, go to **Step 3.**, else – to **Step 10.**

**Step 10.** If $c\geq C$, then the algorithm stops. $S_{00}$ is the best state. Else, in the frames of each $subTM_p$ choose in parallel the neuron $j$ with $\left(t_j\right)_p$ satisfied to $\left(t_j\right)_p=\min\{t_i\,|\,i=\overline{1,N_p}\}, p=\overline{0,(P-1)}$. Using LTMP reduce operations defined by authors, form the set $\left\{j^*,\ \ \Delta E(S_{j^*}),\ \ s_{j^*}\right\}$, where $j^*$ – the index of neuron (in the frames of the whole DTM) changing its state at the current moment, $\Delta E(S_{j^*})$ – the change of DTM energy function value after the neuron $j^*$ has changed its state, $s_{j^*}$ – the new state of neuron $j^*$. Let $S_0 = S_{j^*}$ and $E(S_0) = E(S) + \Delta E(S_{j^*})$, $c = c+1$ and $h = 0$. Go to **Step 6.**

It's worth mentioning that on the **Step 10.** the new state of local energy minimum $E(S_0)$ is set without any auxiliary checks, i.e. is can be worse than the previous $S_0$. Exploiting this technique we exclude stabilization in local energy minimums and expand areas of potential solutions.

## 6 Performance Evaluation

In order to evaluate the performance of constructed DTM the set of experiments on mock-up problems with DTM consisting of $N = 100$, $N = 400$ and $N = 1600$ neurons were done on multi-core cluster. 372 trial solutions were obtained for each mock-up problem depending on the values of < *l, C, β* > parameters of DTM.

We proposed to use an average acceleration as the metric to evaluate the efficiency of DTM. The dependencies of average acceleration on the number of processors for mock-up problems with $N = 100$, $N = 400$ and $N = 1600$ are depicted on Fig. 3. DTM gives a linear acceleration.

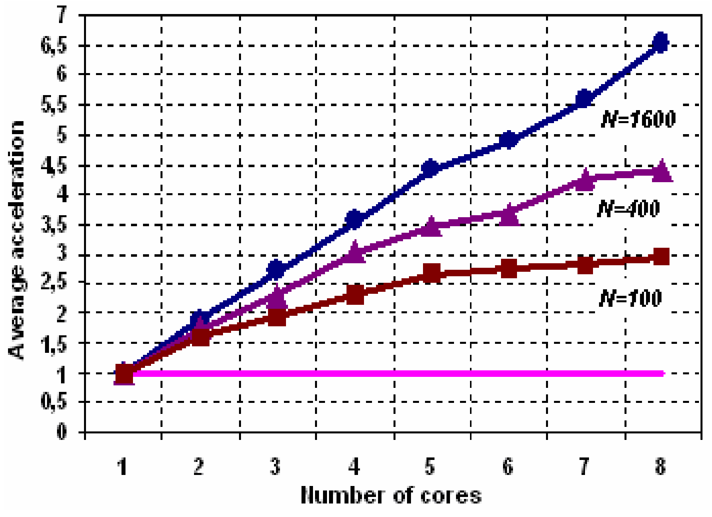

**Fig. 3.** Average acceleration on mock-up problems.

## 7 Conclusion

In this paper we proposed parallel TS algorithm for DDB OLS synthesis problem. The constructed DTM was validated and compared with the sequential TM. As expected, both approaches give the same results with the solutions quality higher than the quality of solutions received by NN-GA-algorithm [1], [3] on average 8,7% and by BBM [8] on average 23,6% on mock-up problem with higher dimension.

It is worth mentioning that during the DTM cycles intensive data communication between processors is carried out in the proposed algorithm. Therefore, we can speak about the significant increasing of DTM performance in compare with its consecutive analogue for the high-dimensional problems. This statement is not contrary to our objectives, because the problem of DDB OLS synthesis is important today in view of high dimensionality.

## References

1. Babkin E., Karpunina M. Comparative study of the Tabu machine and Hopfield networks for discrete optimization problems. Information Technologies'2008. Proc. Of the 14th International Conference on Information and Software Technologies, IT 2008. Kaunas, Lithuania, April 24-25. ISSN 2029-0020. pp. 25-41. (2008)
2. Babkin E., Karpunina M. The analysis of tabu machine parameters applied to discrete optimization problems // Proceedings of 2009 ACS/IEEE International Conference on Computer Systems and Aplications, AICCSA'2009. – May 10-13, 2009. – Rabat, Morocco. – P.153-160. Sponsored by IEEE Computer Society, Arab Computer Society, and EMI, Morocco. (2009) IEEE Catalog Number: CFP09283-CDR. ISBN: 978-1-4244-3806-8. Library of Congress: 200990028. http://www.congreso.us.es/aiccsa2009.
3. Babkin E., Petrova M. Application of genetic algorithms to increase an overall performance of artificial neural networks in the domain of synthesis DDBs optimal structures. Proc. Of The 5th International Conference on Perspectives in Business Informatics Research (BIR 2006) October 6-7, 2006 Kaunas University of Technology, Lithuania. ISSN: 1392-124X Information Techonology and Control, Vol.35, No. 3A. pp.285-294. (2006)
4. Chakrapani J., Skorin-Kapov J. Massively parallel tabu search for the quadratic assignment problem, Annals of Operations Research 41. pp. 327-341. (1993)
5. Fiechter C.-N. A parallel tabu search algorithm for large traveling salesman problems. Discrete Applied Mathematics Vol. 51. ELSEVIER. pp. 243-267. (1994)
6. Garcia B.-L. et al. A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints, Computers Ops Res, Vol.21 No. 9. pp. 1025-1033, (1994)
7. Kant K., Mohapatra P. Internet Data Centers. Computer, Published by the IEEE Computer Society. 0018-9162/04. (2004)
8. Kulba V.V., Kovalevskiy S.S., Kosyachenko S.A., Sirotyuck V.O. Theoretical backgrounds of designing optimum structures of the distributed databases. M.: SINTEG. (1999)
9. Porto Stella C. S., Kitajima Joao Paulo F. W., Ribeiro Celso C. Performance evaluation of a parallel tabu search task scheduling algorithm. Parallel Computing Vol. 26. ELSEVIER. pp. 73-90. (2000)
10. Sun M., Nemati H. R. Tabu Machine: A New Neural Network Solution Approach for Combinatorial Optimization Problems, Journal of Heuristics, 9:5-27, (2003)