
LIVIA SANGEORZAN¹, MIRCEA PARPALEA³, MIHAELA PARPALEA²

¹Department of Computer Science
²Department of Foreign Languages
Transilvania University of Brasov
25, Eroilor, blvd, 500030, Brasov
²National College “Andrei Saguna”
1, Muresenilor, street, 590000, Brasov
ROMANIA
sangeorzan@unitbv.ro, parpalea@gmail.com, parpalea@yahoo.com, ...... http://www.unitbv.ro

Abstract: - The article presents a preflow algorithm for the parametric minimum flow problem working in a parametric residual network with linear lower bound functions of a single parameter. On each of the iterations, the highest-label partitioning preflow-pull algorithm (HLP-P) pulls flow from an active node with the highest distance label over a conditionally admissible arc. After each pull of flow, either the parametric residual capacity of the arc or the parametric deficit of the node becomes zero for at least a subinterval of the parameter values. If the two situations take place in different subintervals, the algorithm is continued in two different parametric residual networks generated by this partitioning pull. The template-like structure of a dialogue act reveals a design where information about the items (part-of-speech) is a multiple section vector with one segment for each of the used part of speech categories. These categories are divided into groups, according to their importance regarding the task, enabling each segment to use its own representations for the words within it.

Key-Words: - Parametric minimum flow, preflow algorithm, fractal partitioning, generative linguistics

1 Introduction
The main feature of neural networks is that knowledge can be represented at a higher level than a single symbol. This method for knowledge representation is known as sub-symbolic. There are some important differences in comparing connectionist techniques (known as neural networks) with traditional, symbolic artificial intelligence [3]. Symbolic artificial intelligence techniques are universal and especially good at searching for solutions by sequences of reasoning steps. Neural networks, on the other hand, are good at perceptual optimizing and common sense tasks but are weak at logical reasoning and search. Symbolic and connectionist artificial intelligence can be seen as complementary approaches.

2 Linguistic rule-based constraints optimisation approach for fractal partitioning of the network
The similarities between parametric networks and neural networks are obvious if taking the parametric minimum flow along every arc as weights $w$ of all incoming arcs. Further more, the contributions to the proportional output of each node might be represented by the sum of all incoming flows based on continuity constraints for all nodes in the network. Choosing linear functions for upper or/and lower bound constrains will result in finding a piecewise linear optimal flow value function approximating the node output function.

2.1 Neural Networks
A biological cell consists of dendrites, a cell body, and an axon. The membrane covering the cell body has a negative electric potential at its resting state. The synapses (narrow gaps) transmit activations between the dendrites. Inputs through excitatory synapses increases, and inputs through inhibitory synapses reduces, the membrane potential. When the potential exceeds a threshold, a pulse is generated which travels through the axon. The more inputs from excitatory synapses a neuron receives, the more frequently it sends an output impulse. The power of neuron comes from its collective behaviour in a network where all neurons are interconnected [1],[2].
Neurons in the network continuously evaluate their output by looking at their inputs, calculating the weighted sum and comparing to a threshold to decide if they should send an output impulse. This is highly complex parallel process whose features can not be reduced to phenomena taking place with individual neurons. A general computational system has a labelled directed graph structure, as presented in Figure 1. Each node performs some simple computations and each
connection transmits a signal from one node to another. All connections are labelled with a number called the connection strength or weight \((w_j)\) that indicates to which extent a signal is augmented or reduced by the connection.

![Figure 1. A general artificial neuron. [4]](image)

The American researcher Frank Rosenblatt’s *perceptron* for pattern classification is an early and one of the most known neural networks (Figure 2). This simple perceptron has two layers of units, the input nodes \((x_1, x_2)\) and one output node, with weighted connections \((w_1, w_2)\) between them. The units take binary activity values (i.e. the pattern to classify has to be encoded as a collection of binary values) and the final output \((y)\) will either be 0 or 1.

![Figure 2. A four-input perceptron with adjustable weights and one output node [4].](image)

Modern networks often use non-linear, proportional output and can thus cope with problems which have more than two answers. Instead of using simple step-functions, more complex functions that are differentiable can be used. The different weights decide the output, \((E = f(w) = f(w_1, w_2, \ldots, w_n))\) and thereby the algorithm adjusts all weights in proportion to their contribution to the output.Popular node functions used in many neural networks are *sigmoid* functions. Digital hardware implementations of the sigmoid function often use piecewise linear approximations.

### 2.2 Constraint-based approaches to language

Constraint-based approaches to language bring a fresh perspective to this important problem because they share a clear analogue with pattern of activation in neural networks. Some of these theories (i.e. Optimality Theory (OT)), take their inspiration from the constraint optimisation that occurs in networks of neurons [5]. As with OT grammaticality judgments, neurons subjected to some input, quickly settle into the state that most satisfies the conflicting weights that constrain the network’s activation. Optimality theory suggests that we should focus upon how the optimisation of conflicting constraints between neurons produces the apparent rule-based behaviour we see in language. In the eyes of many, attacking this foundational problem in Cognitive Science is an over-ambitious task for which we are not yet ready, but the new perspectives brought about by these constraint-based approaches in language suggest otherwise. [6] The principle of constraint satisfaction provides a clear mechanism by which we can construct neural models of high-level, linguistic processes-models which are independent of the complexity of biological neurons, and rely only upon the presence of constraint-satisfaction behaviour.

### 2.3 Flows in parametric networks

The parametric minimum flow problem is an extension of the classical minimum flow problem [8] in which the lower bounds of certain arcs are functions of a parameter \(\lambda\). The parametric minimum flow problem is to compute all minimum flows for every possible value of the parameter. The minimum flow value function in a parametric network is a continuous piecewise linear function of the parameter. Each linear segment of the minimum flow value function between two breakpoints, \(\lambda_k\) and \(\lambda_{k+1}\), corresponds to a cut that remains a maximum cut for any \(\lambda_k < \lambda \leq \lambda_{k+1}\). The approach presented in this article refers to the minimum flow problem in a network with linear lower bound functions of a single parameter \(\lambda\).

#### 2.3.1 Non-parametric flows in networks

Given a capacitated network \(G = (N,A,l,u,s,t)\), let \(n\) being the number of nodes, \(n = |N|\) and \(m\) the number of arcs, \(m = |A|\). The *upper bound* function and the *lower bound* function are nonnegative functions \(u(i,j)\) and \(\ell(i,j)\) associated with each arc \((i,j)\in A\). The network has two special nodes: a source node \(s\) and a sink node \(t\).

A *flow* is a function \(f: A \to \mathbb{R}^+\) satisfying the next conditions:

\[
\sum_{j \in \delta^+(i)} f(i,j) - \sum_{j \in \delta^-(i)} f(j,i) = \begin{cases} v, & i = s \\ 0, & i \neq s, t \\ -v, & i = t \end{cases} \quad (1.a)
\]

for some \(v \geq 0\), where \(v\) is referred to as the *value* of the flow \(f\). Any flow on a directed network satisfying the flow bound constraints, (1.b) for every arc \((i,j)\in A\) is referred to as a feasible flow.

\[
\ell(i,j) \leq f(i,j) \leq u(i,j) \quad \forall (i,j) \in A \quad (1.b)
\]

A cut is a partition of the node set \(N\) into two subsets \(S\) and \(T = N - S\), denoted by \([S,T]\). Let \((S, T)\) denote the set of forward arcs in the cut and \((T, S)\) denote the set of backward arcs. A cut is a *s-t cut* if \(s \in S\) and \(t \in T\).
2.3.2 The minimum flow problem

The minimum flow problem is to determine a flow \( \vec{f} \) for which \( v \) is minimized. The minimum flow problem in a network can be solved in two phases: (1) establishing a feasible flow; (2) from a given feasible flow, establishing the minimum flow. For the first phase see the algorithm presented in [7] and [8].

The residual capacity \( r(i, j) \) of any arc \((i, j) \in A\), with respect to a given flow \( f \), is given by:

\[
r(i, j) = u(j, i) - f(j, i) + f(i, j) - \ell(i, j).
\]

(2)

The capacity of an \( s-t \) cut \( \hat{c}[S, T] \) is defined, for the minimum flow problem, as the sum of the lower bounds of the forward arcs minus the sum of the capacities (upper bounds) of the backward arcs. That is, \( \hat{c}[S, T] = \ell(S, T) - u(T, S) \).

(3)

The \( s-t \) cut with the greatest capacity value among all \( s-t \) cuts is referred to as a maximum cut, \( [\hat{S}, \hat{T}] \).

**Theorem 1** (Min-Flow Max-Cut Theorem) [7], [8]: If there is a feasible flow in the network, the value of the minimum flow from a source \( s \) to a sink \( t \) in a capacitated network with nonnegative lower bounds equals the capacity of the maximum \( s-t \) cut.

In the residual network \( \hat{G}(f) = (N, \hat{A}) \) the distance function, \( \hat{d} : N \to \infty \) is a function from the set of nodes to the nonnegative integers. A distance function is said to be valid if it satisfies the following conditions:

\[
\hat{d}(s) = 0 \quad \text{and} \quad \hat{d}(j) = \hat{d}(i) + 1, \quad \forall (i, j) \in \hat{A}.
\]

(4)

**Definition 1** ([7], [8]): An arc \((i, j) \) in the residual network \( \hat{G}(f) \) is referred to as admissible if it satisfies the condition: \( \hat{d}(j) = \hat{d}(i) + 1 \); otherwise it is inadmissible.

2.3.3 The parametric minimum flow problem

A natural generalization of the minimum flow problem is obtained by making the lower bounds \( \ell(i, j; \lambda) \) linear functions of a single, nonnegative, real parameter \( \lambda \):

\[
\ell(i, j; \lambda) = \ell_0(i, j) - \lambda \cdot \ell(i, j),
\]

(5)

where \( \ell(i, j) \) is a real valued function associated with each arc \((i, j) \in A\), referred to as the parametric part of the lower bound of the arc \((i, j) \). The nonnegative value \( \ell_0(i, j) \) is the lower bound of the arc \((i, j) \) for \( \lambda = 0 \): \( \ell(i, j; 0) = \ell_0(i, j) \) and \( 0 \leq \ell_0(i, j) \leq u(i, j) \).

The parameter \( \lambda \) takes values in the interval \([0, \Lambda]\) where \( \Lambda \) is chosen so that: \( 0 \leq \ell(i, j; \lambda) \leq u(i, j) \), \( \forall (i, j) \in A \); \( \forall \lambda \in [0, \Lambda] \). Therefore the parametric part of the lower bounds, \( \ell(i, j) \) satisfy the following constraints:

\[
\frac{1}{\Lambda} (\ell_0(i, j) - u(i, j)) \leq \ell(i, j) \leq \frac{1}{\Lambda} \ell_0(i, j), \quad \forall (i, j) \in A.
\]

(6)

The parametric minimum flow problem (PMinF) is to compute all minimum flows for every possible value of \( \lambda \in [0, \Lambda] \):

**minimize** \( \tau(\lambda) \) for all \( \lambda \in [0, \Lambda] \)

(7)

with

\[
\sum_{j \in A} f(i, j; \lambda) - \sum_{j \in A} f(j, i; \lambda) = \begin{cases} 
\tau(\lambda), & i = s \\
0, & i \neq s, t \end{cases} \quad (8.a)
\]

\[
\ell(i, j; \lambda) \leq f(i, j; \lambda) \leq u(i, j), \quad \forall (i, j) \in A. \quad (8.b)
\]

The variables \( f(i, j; \lambda) \) of this problem are piecewise linear functions instead of real numbers and the lower bounds \( \ell(i, j; \lambda) \) are linear functions.

On the set of all piecewise linear functions \( f(\lambda) \) an ordering cannot be defined for the whole interval \([0, \Lambda]\) since two piecewise linear functions are not necessarily comparable. Therefore a partitioning \( J_k \) of the interval of the parameter \([0, \Lambda]\) into disjoints subintervals \( J_1 \cup ... \cup J_k = [0, \Lambda] \) with \( J_p \cap J_q = \emptyset, \quad \forall p \neq q \) must be defined so that on each of the subintervals \( J_k \) an ordering to be defined as:

\[
f_i \leq f_j \iff f_i(\lambda) \leq f_j(\lambda), \quad \forall \lambda \in J_k.
\]

(9)

**Definition 2** [9]: A parametric cut partitioning \( \{S_k; J_k\} \) is a finite set of cuts \( \{S_k; T_k\}, \quad k = 1, \ldots, K \) together with a partitioning \( J_k \) of the interval of the parameter \([0, \Lambda]\) into disjoints subintervals so that \( J_1 \cup ... \cup J_k = [0, \Lambda] \) and \( J_p \cap J_q = \emptyset, \quad \forall p \neq q \). The capacity of a parametric \( s-t \) cut partitioning for the minimum flow problem is a piecewise linear function \( \hat{c}[S_k; J_k] \) defined for all \( \lambda \) of every subinterval \( \lambda \in J_k, \quad k = 1, \ldots, K \):

\[
\hat{c}[S_k; J_k] = \sum_{(i,j) \in S_k \cap T_k} \ell(i,j; \lambda) - \sum_{(i,j) \in S_k \cap T_k} u(i,j)
\]

(10)

A parametric \( s-t \) cut for which the subintervals of the parameter values \( \hat{J}_k \) assure that every \( s-t \) cut is a maximum cut \( \hat{S}_k; \hat{T}_k \) for all \( \lambda \in \hat{J}_k \) is referred to as a parametric maximum \( s-t \) cut, \( \{\hat{S}_k; \hat{T}_k\} \) for the whole interval of the parameter values, \([0, \Lambda]\). A parametric maximum cut \( \{\hat{S}_k; \hat{T}_k\} \) is a set of maximum cuts \( \{\hat{S}_k; \hat{T}_k\} \) and \( \hat{c}[\hat{S}_k; \hat{T}_k] \) for all \( \lambda \) of every subinterval \( \hat{J}_k, \quad k = 1, \ldots, K \).

**Theorem 2** (Parametric Min-Flow Max-Cut Theorem) [9]: If there is a feasible flow in the parametric network, the value function of the parametric minimum flow from
a source \( s \) to a sink \( t \) in a capacitated network with parametric lower bounds equals the capacity of the parametric maximum \( s-t \) cut.

Let \( \mathbf{f}(\lambda) = (f(i,j;\lambda)) \) be a vector of flow functions defined on the interval \([0,\Lambda]\). Supposing that an arc \((i,j) \in A\) carries a flow \( f(i,j;\lambda) \), the existing flow can be reduced either by pulling the flow \( f(i,j;\lambda) - \ell(i,j;\lambda) \) from node \( j \) to node \( i \), by only pushing the flow \( u(j,i) - f(j,i;\lambda) \) from \( j \) to \( i \) along the arc \((j,i)\). The parametric residual capacity \( \hat{r}(i,j;\lambda) \) of any arc \((i,j) \in A\), with respect to a given flow \( \mathbf{f}(\lambda) \), is given by:

\[
\hat{r}(i,j;\lambda) = u(j,i) - f(j,i;\lambda) + f(i,j;\lambda) - \ell(i,j;\lambda).
\]

For a network \( G = (N, A, \ell(\lambda), u, s, t) \) and a feasible solution \( \mathbf{f}(\lambda) \), the network denoted by \( \hat{G}(\mathbf{f}(\lambda)) = (\hat{N}, \hat{A}) \)

\[
\hat{N} = N \quad \text{and} \quad \hat{A} = \{ \text{the set of arcs consisting only of arcs with } \hat{r}(i,j;\lambda) > 0 \text{ for at least a subinterval of } [0,\Lambda] \}
\]

is referred to as the parametric residual network with respect to the given flow \( \mathbf{f}(\lambda) \) for the parametric minimum flow problem.

The sets:

\[
\hat{I}(i,j) = \{ \lambda | \hat{r}(i,j;\lambda) > 0 \} \quad \text{for } (i,j) \in \hat{A}, \quad \hat{I}(i,j) \subseteq [0,\Lambda]
\]

describe subintervals of \([0,\Lambda]\), \( \hat{I}(i,j) \subseteq [0,\Lambda] \) where a decreasing of flow along an arc \((i,j) \in \hat{G}(\mathbf{f}(\lambda)) \) is possible, based on \( \mathbf{f}(\lambda) \). If an arc \((i,j) \) doesn’t belong to \( \hat{G}(\mathbf{f}(\lambda)) \) then \( \hat{I}(i,j) = \emptyset \).

The parametric deficit of a node \( i \in N \) is defined as:

\[
\hat{d}(i) = \sum_{(i,j) \in A} f(i,j;\lambda) - \sum_{(j,i) \in A} f(j,i;\lambda).
\]

The sets:

\[
\hat{I}(j) = \{ \lambda | \hat{d}(j;\lambda) < 0 \} \quad \text{for } j \in N - \{s,t\},
\]

describe subintervals of \([0,\Lambda]\), \( \hat{I}(j) \subseteq [0,\Lambda] \) where the deficit of node \( j \) is negative.

An arc \((i,j) \in \hat{A} \) in the parametric residual network \( \hat{G}(\mathbf{f}(\lambda)) \) is referred to as conditionally admissible arc if \( \hat{d}(j) = \hat{d}(i) + 1 \) and \( \hat{I}(i,j) \cap \hat{I}(j) = \emptyset \); otherwise it is conditionally inadmissible.

A conditional decreasing directed path \( \hat{P}(\lambda) \) in \( \hat{G}(\mathbf{f}(\lambda)) \) is a directed path \( \hat{P} \) from the source node \( s \) to the sink node \( t \) such that:

\[
\hat{I}(\hat{P}) = \bigcap_{(i,j) \in \hat{P}} \hat{I}(i,j) \neq \emptyset,
\]

i.e. including only conditionally admissible arcs.

### 3 Network partitioning preflow-pull algorithm

The proposed algorithm for the parametric minimum flow problem uses a fractal-like approach [10]. A non-canceling pull of flow from any node of the network might leave the node unbalanced for only a subinterval of the parameter. We call this situation a partitioning pull. Like in all fractal approaches a partitioning pull is followed by separating the problem into disjoints subintervals allowing the algorithm to continue after the same rules independently on each of the partitioned subintervals. The idea of the approach derives both from the rules and recursion of generative linguistics on one hand and from the collective behaviour of neurons in a network on the other hand.

#### 3.1 Highest-label partitioning preflow-pull algorithm (H-L P P-P) for the parametric minimum flow problem

The first phase of finding a parametric minimum flow consists in establishing a feasible flow in a nonparametric network \( G = (N, A, \ell, u, s, t) \) obtained from the initial network \( G = (N, A, \ell(\lambda), u, s, t) \) by only modifying the parametric lower bounds as follows:

\[
\ell'(i,j) = \max \{ \ell(i,j;\lambda) | \lambda \in [0,\Lambda] \}, \quad \text{i.e.} \quad \ell'(i,j) = \ell_0(i,j) \quad \text{for } \mathbf{L}(i,j) \geq 0 \quad \text{and} \quad \ell'(i,j) = \ell_0(i,j) - \Lambda \cdot \mathbf{L}(i,j) \quad \text{for } \mathbf{L}(i,j) < 0.
\]

In the second phase, the algorithm maintains a set, \( L \) of active nodes as a priority queue. In the **initialisation** step of the algorithm, the flow is set to its lower bound value for every arc, \( f(i,t;\lambda) := \ell(i,t;\lambda) \) and all nodes \( i \neq s \) which become active nodes are added to the priority queue \( L \) with priorities \( \hat{d}(i) \). The active nodes are then removed from the priority queue in the descending order of their priorities. For an active node \( j \in \hat{G}(\mathbf{f}(\lambda)) \), if there exists an conditionally admissible arc \((i,j)\), the flow will be pulled on this arc and if \( i \neq s \), \( i \neq t \) and \( i \in L \), node \( i \) will be added to the priority queue \( L \) with the priority \( \hat{d}(i) \); otherwise the node \( j \) will be relabelled so that at least one conditionally admissible arc to be created and node \( j \) is added to \( L \) with it’s new priority \( \hat{d}(j) \). The algorithm terminates when the queue of active nodes is empty. A pull of flow from node \( j \) to node \( i \) is referred to as a **canceling pull** if it deletes the arc \((i,j)\) from the residual network; otherwise it is a **non-canceling pull**. For any node in \( \hat{G}(\mathbf{f}(\lambda)) \), the expressions: **active node** and **balanced node** holds only for subintervals of the parameter values. While both the parametric residual capacity \( \hat{r}(i,j;\lambda) \) of any arc, \((i,j) \in A \) and the
parametric deficit \( \hat{d}(i; \lambda) \) of any node \( i \in N - \{s,t\} \) are piecewise linear functions, cancelling or non-cancelling pulls are defined only for certain subintervals of the parameter values.

A non-cancelling pull of flow from a node \( j \in N - \{s,t\} \) along an arc \((i,j) \in A\) in a subinterval \( J_p = [\lambda_{p-1}, \lambda_p] \subseteq [0, \Lambda] \) which leaves the node \( j \) unbalanced is referred to as a partitioning pull.

Whenever the algorithm performs a partitioning pull in \( \hat{G}_p(f, \lambda) \), a new partitioning of \( J_p \) in at most two subintervals \( I_{p1} \) and \( I_{p2} \), with \( I_{p1} \cup I_{p2} = J_p \) and \( I_{p1} \cap I_{p2} = \emptyset \) may take place. Let \( I_{p1} \) be the subinterval inside which the partitioning pull balances the node \( j \), i.e. \( I_{p1} = [\lambda - \hat{d}(i; \lambda) \leq \hat{r}(i,j; \lambda)] \). If \( I_{p2} \neq \emptyset \) then, as on every subinterval \( I_p \) both \( \hat{r}(i,j; \lambda) \) and \( \hat{d}(i; \lambda) \) are linear functions of \( \lambda \), the partitioning pull generates two parametric residual networks, \( \hat{G}_{p1}(f, \lambda) \) for \( \lambda \in I_{p1} \) and \( \hat{G}_{p2}(f, \lambda) \) for \( \lambda \in I_{p2} \), so that node \( j \) is balanced in \( \hat{G}_{p1}(f, \lambda) \) and active in \( \hat{G}_{p2}(f, \lambda) \) while arc \((i,j)\) does not belong to \( \hat{G}_{p2}(f, \lambda) \) since \( \hat{r}(i,j; \lambda) = 0 \). The algorithm then continues separately in each of the parametric residual networks and for each of the two subintervals. Under these observations, the pull/relabel procedure of the non-parametric HL-PP algorithm is replaced with a recursive call of a partitioning pull\((L,J)\) procedure.

**HL-PPF Algorithm;**

begin
  initialization;
  partitioning pull\((L,J)\);
end;

**Theorem 3:** If there is a feasible flow in the network \( G=(N,A,f,u,s,t) \), the Highest-label partitioning preflow-pull algorithm computes correctly a minimum flow.

**Proof:** The proof of the proposed theorem follows from the correctness of the HL preflow-pull algorithm for each of the subintervals of the parameter values.

### 3.2 Complexity issues

A breakpoint is a place where the slope of the piecewise linear minimum flow value function is changing. In the worst case the number of breakpoints may be exponential in the size of the problem. The example originates from the pathological graph of Zadeh which was used to show that the complexity may be exponential [11]. The Highest-label partitioning preflow-pull algorithm overcomes this inconvenient by using the multi-thread parallel implementation of a non-parametric algorithm [12]. The main idea of this implementation, deriving from the collective behaviour in a network where all neurons are interconnected, is to assign a processor to each newly generated subinterval \( J_p \) which will carry out the problem forward from the current configuration of the problem. For each of the new generated subintervals a copy of the current distance labels values is generated so that they can be independently modified in the further parallel evolution of the algorithm. This approach is illustrated in Figure 3.

![Figure 3. The running time of the HL P-P-P algorithm](image_url)

**Theorem 4:** The Highest-label partitioning preflow-pull algorithm solves the parametric minimum flow problem in \( O(n^2 m^{1/2} + Kn) \) time.

**Proof:** The complexity of the non-parametric HL preflow-pull algorithm is \( O(n^2 m^{1/2}) \), (For details see [7]). The HL -partitioning preflow -pull algorithm generates new copies of distance label values every time a breakpoint occurs, i.e. copying distance labels takes \( O(Kn) \) time where \( K \) is the number of breakpoints. Thus, the total complexity of the algorithm is \( O(n^2 m^{1/2} + Kn) \).

### 4 Neural Networks and Linguistic Theory

The concept which has proved most useful in the description of German word order has become known under the name of Functional Sentence Perspective (FSP). Its principal idea is that information is not transmitted in random order, but that the speaker seeks to give his information to his interlocutor in portions, normally starting from what he assumes is common to both (the THEME, topic) and proceeding to what he regards as important new information (the RHEME, comment) [13].

Before the work of Michael Kipp [14] the problem of finding appropriate dialogue acts for given utterances in large speech corpora was solved through symbolic,
decision tree and statistical approaches. When determining a dialogue act, information about the items part-of-speech (POS) is valuable, i.e. the final design of the system comprises a multiple section vector with one segment for each of the used POS-categories. The POS-categories are divided into groups according to their importance regarding the task. This enables each POS-segment to use its own representations for the words within it. In a POS of high importance each word is represented by one distinct component. Within a medium-important POS all words are represented by the binary value indicating their position in the word list of the category. Words with low importance were grouped to a single vector component lacking information about the specific tokens, the interesting information being the category itself.

One of the most widely used neural network algorithms is the Self-Organizing Map (SOM) [3]. The learning process is competitive and unsupervised, i.e. no desired output is involved in the training of the net. The first layer of a SOM is the input layer and each node in the second layer is the winner for all input vectors in a region of input space. Each $j^{th}$ node in the second layer is described by the vector of weights to it from the input nodes: $(w_j) = (w_{j1}, w_{j2}, \ldots, w_{jn})$. Weight vectors have the same dimension as input vectors and can thus be compared to these by a distance measure estimating the similarity of them. A winner is the node whose weight vector $(w_1, w_2, \ldots, w_n)$ is nearest to an input pattern. The weight vector associated with each node is as near as possible to all input samples for which the node is the winner. At the same time it is as far away as possible to very different samples.

5 Conclusion

The minimum flow problem on parametric networks turns out to be an important scenario in practice since the complexity of its solving algorithm can be reduced to that of the equivalent non-parametric algorithm considering the approach of fractal partitioning of the network on one hand and the rules and recursion of generative linguistics and the collective behaviour of neurons in a network on the other. The weaknesses and strengths of the parametric minimum flow problem can be further developed taking advantage of the neural network approach. It seems that the use of artificial neural network in language models offers a promising methodology to explore linguistic resources of various kinds. There is a lot of interesting work in this active field, as morpho-syntactic disambiguation, developed full-text analysis, information retrieval and also interpretation of sign languages.

References:
[4] Negishi, M. Everything that Linguists have Always Wanted to Know about Connectionism, Department of Cognitive and Neural Systems, Boston University, 1998