

Spreading Activation Simulation with Semantic Network Skeletons

Kerstin Hartig

Technische Universität Berlin, Germany
Email: kerstin.hartig@tu-berlin.de

Thomas Karbe

aklamio GmbH, Berlin, Germany
Email: thomas.karbe@aklamio.com

Abstract—Spreading activation algorithms are a well-known tool to determine the mutual relevance of nodes in a semantic network. Although often used, the configuration of a spreading activation algorithm is usually problem-specific and experience-driven. However, an excessive exploration of spreading behavior is often not applicable due to the size of most semantic networks. A semantic network skeleton provides a comprised summary of a semantic network for better understanding the network’s structural characteristics. In this article, we present an approach for spreading activation simulation of semantic networks utilizing their semantic network skeletons. We show how expected spreading activation behavior can be estimated and how the results allow for further effect detection. The appropriateness of the simulation results as well as time-related advantages are demonstrated in a case study.

Keywords—Spreading Activation; Simulation; Semantic Network; Semantic Network Skeleton; Information Retrieval.

I. INTRODUCTION

Semantic networks are a well-established technique for representing knowledge by nodes and connected edges. Widely used in many areas, their utilization is object of scientific research itself such as creating, transforming, and searching these networks. Such networks often tend to be large in order to profit from the semantic expressiveness of detailed contained knowledge. Despite their graphic structure, increasing network size might hinder their comprehensibility, and their usage might become cumbersome and time-consuming.

Semantic network skeletons are a tool designed for better understanding a semantic network’s structural properties [1]. A skeleton summarizes basic characteristics of a semantic network and, thus, focuses on a few essential pieces of information. Its abstracted and comprised character enables various analyses.

One essential operation when using semantic networks is retrieving information, e.g., with semantic search algorithms such as spreading activation. Spreading activation algorithms are a long-known tool to determine relevance of nodes in a semantic network. Originally from psychology, they have been used in many other application areas, such as databases, artificial intelligence, biology, and information retrieval [2].

All spreading activation algorithms follow a basic pattern: chunks of activation are spread gradually from nodes to neighboring nodes, which marks receiving nodes as being relevant to a certain degree. However, practically, each known implementation differs in many details, such as the amount and distribution of activation. Whether a specific configuration for such an algorithm leads to useful results depends largely on two factors: the problem to be solved by spreading and the structure of the underlying semantic network. Although there are many working examples of such algorithms, until

now there are almost no guidelines on how to achieve a good configuration. Knowledge about effects and their causes facilitates pre-configuration analyses in order to optimize the settings to retrieve the desired effects. Since semantic networks tend to be very large, an excessive examination of spreading behavior with a multitude of configuration settings can be a time-consuming task.

Therefore, we propose to utilize the comprised structural summary of a semantic network skeleton for spreading activation simulation. In this article, we aim to gain insights on the spreading activation behavior on a semantic network by simulating spreading activation on its network’s skeleton. We present a framework for spreading activation simulation that supports detailed observations of two basic properties. First, we observe the activation strength, i.e., the pulsewise development of activation values within a network. Second, we track the spreading strength, i.e., the number of nodes that are activated, which is a measure for activation saturation in the semantic network. The simulation results can reveal desired and undesired effects and allow for further pre-configuration analyses such as sink detection.

In Section II, we will give a short summary about semantic networks and spreading activation. In Section III, we provide a formal framework for spreading activation in semantic networks, and introduce an extension that we refer to as spreading modes. Section IV is dedicated to semantic network skeletons formally and visually. In Section V, we introduce our spreading simulation approach formally, and provide examples for the simulation steps. Section VI is dedicated to the evaluation of the presented simulation approach. We will show that simulation results match their corresponding spreading results at an appropriate average, and we present time-related advantages. We finish the article with conclusions and an outlook on future research potentials regarding spreading activation simulation and semantic network skeletons.

This article is an extension of a previous paper [1], where we introduced the concept of semantic network skeletons. In this article, we extend this approach by showing how skeletons can be used for simulating spreading activation. We furthermore show that the simulation results are predictors for the actual spreading activation on the original network.

II. BASICS AND RELATED WORK

We simulate spreading activation as semantic search technique on semantic networks. Therefore, we shed some light on the underlying concepts.

A. Semantic Network

Historically, the term semantic network had its origin in the fields of psychology and psycholinguistics. Here, a semantic

network was defined as an explanatory model of human knowledge representation [3][4]. In such a network, concepts are represented by nodes and the associations between concepts as links. Generally, a semantic network is a graphic notation for representing knowledge with nodes and arcs [5][6]. Notations range from purely graphical to definitions in formal logic.

Technically, among others semantic networks can be described by the Resource Description Framework (RDF) and RDF Schema (RDFS). The RDF data model [6] is defined to be a set of RDF triples whereas each triple consists of a subject, a predicate and an object. The elements can be Internationalized Resource Identifiers (IRI), blank nodes, or datatyped literals. Each triple can be read as a statement representing the underlying knowledge. A set of triples forms an RDF Graph, which can be visualized as directed graph, where the nodes represent subject and object and a directed edge represents the predicate [6].

B. Spreading Activation

Spreading activation, like semantic networks, has a historical psychology and psycholinguistic background. It was used as a theoretical model to explain semantic memory search and semantic preparation or priming [3][4][7].

Over the years, spreading activation evolved into a highly configurable semantic search algorithm and found its application in different fields. In a comprehensive survey, Crestani examined different approaches to the use and application of spreading activation techniques, especially in associative information retrieval [2]. Spreading activation is capable of both identifying and ranking the relevant environment in a semantic network.

1) *Processing*: The processing of spreading activation is usually defined as a sequence of one or more iterations, so-called pulses. Each node in a network has an activation value that describes its current relevance in the search. In each pulse, activated nodes spread their activation over the network towards associated concepts, and thus mark semantically related nodes [2]. If a termination condition is met, the algorithm will stop. Each pulse consists of different phases in which the activation values are computed by individually configured activation functions. Additional constraints control the activation and influence the outcome considerably. Moreover, constraint-free spreading activation leads to query-independent results [8]. Fan-out constraints limit the spreading of highly connected nodes because a broad semantic meaning may weaken the results. Path constraints privilege certain paths or parts of them. Distance constraints reduce activation of distant nodes because distant nodes are considered to be less associated to each other. There are many other configuration details such as decays, thresholds, and spreading directions.

2) *Application Areas*: Álvarez et al. introduced the OntoSpread Framework for the application and configuration of spreading activation over RDF Graphs and ontologies [9]. They use their framework for retrieving recommendations, e.g., in the medical domain [10]. Grad-Gyenge et al. use spreading activation to retrieve knowledge graph based recommendations for email remarketing [11]. Crestani et al. applied constrained spreading activation techniques for searching the World Wide Web [12]. An approach for Semantic Web trust management utilizes spreading activation for trust propagation [13]. Another area of application is the semantic desktop, which aims at transferring semantic web technologies to the users desktop.

Schumacher et al. apply spreading activation in semantic desktop information retrieval [14].

3) *Configuration*: A challenge mentioned in spreading activation related research is the tuning of the parameters, e.g., values associated with the different constraints as well as weighting or activation functions. For evaluation of the prototype WebSCSA (Web Search by Constrained Spreading Activation) in [12], values and spreading activation settings are identified experimentally, empirically, or partly manually according to the experiments requirements. Álvarez et al. state that a deep knowledge of the domain and the semantic network is necessary and domain-specific customization configuration is needed [9]. In a case study from the medical systems domain, they emphasize the need for automatic support for proper configuration selection, e.g., by applying learning algorithms [10]. It is a known fact that spreading activation configuration has a huge impact on the quality of the spreading results. Currently, there exists no systematic approach for the determination of proper configuration settings. Moreover, not even guidelines for the appropriate configuration are available to potential users. There is a lack of systematic analyses of the impact and interaction of different settings and parameters. The simulation approach presented in this article aims at facilitating such analyses in order to gain helpful insights and support appropriate configurations.

C. Simulation

The common idea of simulation is to imitate the operation of real-world processes or systems over time [15]. Simulation is applied in different domains, such as traffic, climate, medical science, or engineering [16]. Usually, simulation is performed on a model, which is an approximation of the item to be simulated, because real world is often too complex [16]. Simulation on this model facilitates repeated observation of specific events, which can be utilized for analyses in order to draw conclusions.

In this article, we aim at simulating the behavior of an algorithm under different configurations on a specific data structure, i.e., very large semantic networks. Here, the simulation model is the semantic network skeleton, which is used to approximate the behavior of the spreading algorithm on the underlying semantic network. Another approach uses simulation of algorithms, for example, in the context of signal processing [17]. Here, simulation was performed on a MATLAB model in order to optimize parameters such as sampling rates or filter designs before implementing the algorithm in hardware. In [18], the authors describe the necessity of tuning coordination algorithms for robots and agents as well as the challenge of finding proper configurations due to a large configuration space. They use simulation to collect data to train neural networks in order to optimize performance data.

Our approach also aims at tuning configuration parameters. However, our simulation method does not target direct configuration optimization, but indirectly targets approximated results to better understand the beforementioned interdependence.

III. SPREADING ACTIVATION IN SEMANTIC NETWORKS

Spreading activation based algorithms follow a common principle but may vary in detail. Therefore, we present a framework that we will use as foundation for the simulation approach introduced in Section V. We focus on the basic pure spreading approach and describe three well-established

constraints. Additionally, we introduce an extension that we refer to as spreading modes.

A. Semantic Networks

Let L be a set of labels. The semantic network G (here also source network) is a directed labelled multigraph and defined by

$$G = (N, E, s, t, l, \omega)$$

where

- N is a non-empty set of nodes,
- E is a set of edges,
- $s : E \rightarrow N$ is the edge source mapping,
- $t : E \rightarrow N$ is the edge target mapping,
- $l : N \cup E \rightarrow L$ is the labelling,
- $\omega : E \rightarrow \mathbb{R}$ is the edge weight mapping.

B. Basic Spreading Activation Functions

The principle of spreading activation is a combination of pulse-wise computations of the three spreading activation functions. For $n \in N$, $e \in E$, and spreading pulse $p \geq 0$:

- the output function $out : N \times E \times \mathbb{R} \rightarrow \mathbb{R}$ determines the state of output activation $o_n^{(p)}$ for node n at pulse p ,
- the input function $in : N \times E \times \mathbb{R} \rightarrow \mathbb{R}$ determines the state of input activation $i_{n,e}^{(p)}$ for node n via edge e at pulse p ,
- the input function $in : N \times \mathbb{N} \rightarrow \mathbb{R}$ determines the state of input activation $i_n^{(p)}$ for node n at pulse p , and
- the activation function $act : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ determines the activation level $a_n^{(p)}$ for node n at pulse p .

These functions are computed in each spreading pulse $p \geq 0$, where $p = 0$ denotes the initial state and, therefore, the starting point of the algorithm. The computations in each pulse follow a specified order. A new pulse starts with calculating the output activation utilizing the latest activation level from the previous pulse. Subsequently, the input activation of all nodes are determined from the output activation, which is finally used for calculating the new activation level.

A node is defined to be activated as soon as it received any activation value in a spreading activation step. In subsequent steps, the strength of a node's activation may increase, but a node can not be deactivated.

On Spreading Directions: Spreading activation can consider or ignore the direction of edges. Neglecting edge directions is more intuitive since the direction solely reflects the reading direction of an edge's property. Redirecting an edge does not change the semantic meaningfulness between the connected nodes, e.g., x *hasMalfunction* y can be read as y *isMalfunctionOf* x . Spreading activation based algorithms utilize semantic relatedness within semantic networks symbolized by their structure. Therefore, we present spreading activation functions that ignore edge directions and spread activation over all edges connected to a node, referred to as *undirected spreading* in the remainder of this article. However, the presented algorithms can easily be adapted to adhere to the assigned edge property directions.

Output activation function: The state of output activation $o_n^{(p)}$ for node $n \in N$, $e \in E$ at each pulse $p > 0$ is determined by the output function:

$$o_n^{(p)} = out(n, e, a_n^{(p-1)}), \quad (1)$$

where in pure undirected spreading

$$out(n, e, a) := \begin{cases} a & \text{if } s(e) = n \vee t(e) = n, \\ 0 & \text{else.} \end{cases} \quad (2)$$

Input activation function: The input activation $i_{n,e}^{(p)}$ for nodes $n, m \in N$ via edge $e \in E$ at each pulse $p > 0$ is determined by the input function:

$$i_{n,e}^{(p)} = in(n, e, o_m^{(p)}), \quad (3)$$

where in pure undirected spreading

$$in(n, e, o) := \begin{cases} o_{s(e)} \cdot \omega(e) & \text{if } t(e) = n, \\ o_{t(e)} \cdot \omega(e) & \text{if } s(e) = n, \\ 0 & \text{else.} \end{cases} \quad (4)$$

The consolidated input activation of a node n received via all edges can be combined:

$$i_n^{(p)} = in(n, p), \quad (5)$$

where

$$in(n, p) := \sum_{e \in E} i_{n,e}^{(p)}. \quad (6)$$

Activation function: The activation level $a_n^{(p)}$ for node $n \in N$ describes the current assigned activation value at each pulse $p \geq 0$, where $a_n^{(0)}$ denotes the initial activation level of n . For $p > 0$, the activation level is determined by the activation function:

$$a_n^{(p)} = act(i_n^{(p)}, a_n^{(p-1)}), \quad (7)$$

where in pure spreading the input activation is added directly to the latest activation level of node n :

$$act(i, a) := i + a. \quad (8)$$

The definition of spreading activation functions can be versatile. Additional computations can be applied such as normalization functions. Here, only basic spreading activation functions are presented. However, we apply some well-known constraints and several newly defined spreading modes that will be integrated in the output activation function.

Constraints: There are various known spreading configuration parameters, also called constraints, applied in several applications and presented in information retrieval research work [2]. Constraints allow for additional control of the three kinds of activation functions. In this article, we chose three known constraints, and show how we include them into our spreading approach. Other parameters are known and can be applied in the same manner.

First, the *activation threshold* controls whether or not nodes with only very low activation values under a specified threshold are excluded from spreading their activation in the specific pulse. Since the activation value is a measure for the relevance of a node, it impedes semantically non-relevant nodes to contribute in the spreading process. Here, the activation threshold is controlled by $\tau : N \rightarrow \mathbb{R}$. If no activation threshold is applied, $\tau = 0$.

Second, the *fan-out* constraint provokes the splitting of outgoing activation due to the assumption that highly connected nodes have a broad semantic meaning and, therefore, may weaken the informative value of the spreading result. This punishes highly connected nodes by reducing the activation value to be passed, mostly by splitting the activation equally to those nodes. Whether or not the fan-out constraint is applied is controlled by the boolean parameter *fanout*.

Third, the *pulse decay* decreases the transported activation over the amount of activation steps, i.e., pulses. On the one hand, this punishes distant nodes. On the other hand, it curbs the overall activation distribution over the iterations. The pulse decay is controlled by the decay factor d . If the pulse decay constraint is not applied, $d = 1$.

The three presented constraints affect the level of output activation and must be considered in the output activation function. The constraints-aware output functions are defined as follows:

The output function with a pulse decay and a decay factor d is defined by:

$$out(n, e, a) := \begin{cases} d \cdot a & \text{if } s(e) = n \vee t(e) = n, \\ 0 & \text{else.} \end{cases} \quad (9)$$

The output function with an activation threshold τ :

$$out(n, e, a) := \begin{cases} a & \text{if } s(e) = n \vee t(e) = n, a \geq \tau, \\ 0 & \text{else.} \end{cases} \quad (10)$$

The output function with a fan-out constraint that equally divides the available activation by the outgoing edges:

$$out(n, e, a) := \begin{cases} \frac{a}{deg(n)} & \text{if } s(e) = n \vee t(e) = n, \\ 0 & \text{else,} \end{cases} \quad (11)$$

with the degree of a node $n \in N$ when ignoring edge directions:

$$deg(n) := |\{e \in E | s(e) = n \vee t(e) = n\}|. \quad (12)$$

Constraints can be combined in the output function, e.g., choosing all three presented constraints results in the output activation:

$$out(n, e, a) := \begin{cases} d \cdot a & \text{if } s(e) = n \vee t(e) = n, \\ & a \geq \tau, \neg fanout, \\ \frac{d \cdot a}{deg(n)} & \text{if } s(e) = n \vee t(e) = n, \\ & a \geq \tau, fanout, \\ 0 & \text{else.} \end{cases} \quad (13)$$

C. Extension - Spreading Activation Modes

The processing of spreading activation based algorithms in information retrieval applications can follow manifold conceptions of how the activation is supposed to spread over networks. Mostly, the processing of spreading activation is based on the assumption that in each pulse every activated node of the network is allowed to spread its activation (or part of it) to neighbor nodes. In that case, whether or not a node indeed provides output activation only depends on restrictions coming from additional constraints such as an activation threshold.

However, we see potential in an extended and more distinguished treatment of nodes by deciding whether a node gets permission to spread (and receive) activation. Therefore, we distinguish between various so-called spreading modes. Such modes define spreading rules and, therefore, control the paths taken during the activation process. Moreover, we can distinguish between the edges that transport activation, e.g., a node does not necessarily have to be allowed to spread via each of its connected edges. Formally, each spreading mode affects the output activation by the spread permission function φ . The spread permission function can be applied to any output activation function. Here, we introduce three intuitive modes. The mode- and constraint-aware output activation function is defined by

$$out(n, e, a, \varphi) := \varphi \cdot out(n, e, a). \quad (14)$$

The presented modes ignore edge directions since output functions carry this information already before mode-aware extension. We distinguish between the following spreading modes.

1) **Basic Mode:** As mentioned before, the *basic spreading mode* allows each node to spread activation to all neighbors in each pulse, regardless of the edge's directions. Of course, only activated nodes with activation values greater than zero can generate an amount of output activation for spreading. However, this is controlled by the output function. The permission function is defined as:

$$\varphi(n, e) := 1. \quad (15)$$

This means that each node is theoretically allowed to spread via all edges. Practically, a node is usually not connected with all edges. Note, that for a non-connected edge e of node n , the spreading permission is repealed by the non-existing output activation $o_{n,e}$.

2) **Recent Receiver Mode:** Another mode solely allows nodes that were receivers of activation in the last pulse to spread activation to their neighbor nodes. The permission function is defined as:

$$\varphi(n, e, p) := \begin{cases} 1 & \text{if } p = 0, \\ 1 & \text{if } i_n^{(p-1)} > 0, p > 0, \\ 0 & \text{else.} \end{cases} \quad (16)$$

3) **Forward Path Mode:** Another mode evolves when nodes may not directly spread back to the nodes they just received activation from. The permission function is defined as:

$$\varphi(n, e, p) := \begin{cases} 1 & \text{if } p = 0, \\ 1 & \text{if } i_{n,e}^{(p-1)} = 0, p > 0 \\ 0 & \text{else.} \end{cases} \quad (17)$$

In Figure 1, the spreading paths of the three presented modes are depicted. Not each activated node must necessarily be permitted to spread to neighbor nodes. While in basic mode each activated node is permitted to spread over all connected edges, the figure reveals that in recent receiver mode the spreading permission of node A and D will pulse-wise alternate. In forward path mode, we observe that spreading back is only permitted on circular paths, e.g., A-B-C-A, but not

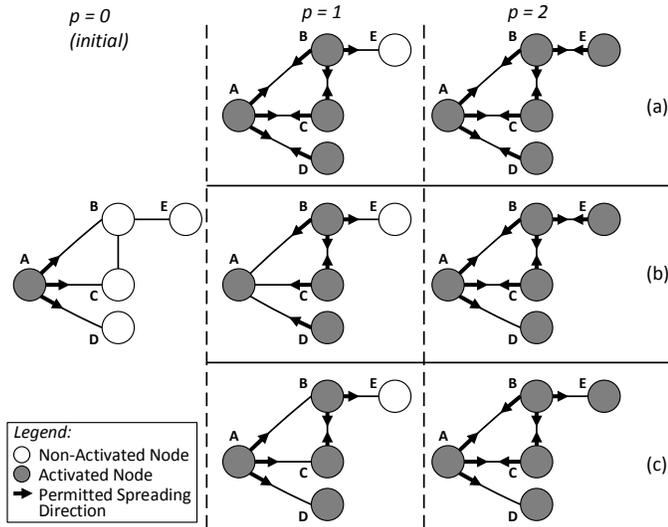


Figure 1. Mode-Aware Permitted Spreading Behavior in three Activation Pulses in a) Basic Mode b) Recent Receiver Mode c) Forward Path Mode.

directly A-D-A. Here, circular paths are privileged since they might incorporate special semantic meaning whereas back-spreading of linear paths is impeded. Different modes are a measure that may impede extended oscillated spreading by means of a sophisticated spreading permission system.

In contrast to constraints, spreading modes can not be combined. Exactly one mode must be chosen, where the basic mode can be seen as a default mode.

IV. SEMANTIC NETWORK SKELETON

As stated before, proper configuration of a spreading activation algorithm is a challenging task. One important influencing factor for a good configuration is the structure of the underlying semantic network. Often however, semantic networks tend to be very large, and therefore hard to comprehend.

We propose a tool called *semantic network skeleton*, introduced in [1], which is supposed to summarize the structure of a semantic network. Therefore, using a skeleton shall make it easier to comprehend their structural properties and draw conclusions for configurations.

A. Skeleton Introduction

A skeleton of a semantic network is a directed graph that has been derived from a semantic network. We will call the semantic network from which the skeleton has been derived the *source (network)*.

Generally spoken, the skeleton shall represent the semantic structure of the source. Therefore, similar nodes and edges are grouped and represented by single node representatives and edge representatives in the skeleton. Thus, the skeleton hides all the parts of the source which are similar, and it makes the structural differences in the network more explicit.

Often, a semantic network contains also nodes and edges that carry little semantic value and therefore should be ignored by a spreading activation algorithm. An example from the RDF Specification are blank nodes, which by definition carry no specific meaning. Therefore, before creating a skeleton from a source, one first has to define the *semantic carrying* set of nodes and edges. This choice is very problem-specific, and

therefore cannot be generalized. We call the semantic carrying subnetwork of the source the *spread graph*.

Since the skeleton is based on the spread graph, it represents only semantic carrying nodes and edges. The skeleton usually contains three types of node representatives: classes, instances, and literals. Since the relationships between instance node representatives carry the most structural information about the semantic network, we call this part the *skeleton core*.

B. Types of Semantic Network Skeletons

We distinguish between two types of skeletons regarding their completeness and detail level: the maximum and the effective skeleton of a network.

A *maximum skeleton* contains all potential nodes and relations of the source. It is comparable with a UML class diagram in the sense that it shows everything that is theoretically possible in that network. However, it does not transport any information about the actual usage of classes/instances in the source network. Therefore, the maximum skeleton might contain nodes and relationships that have never been instantiated in the source.

An *effective skeleton* represents the structure of a specific instance of a semantic network. Therefore, it contains only nodes and relations that are actually part of the source network. This means that a class that is part of an RDF schema, but that has not been instantiated in a concrete instance of that RDF schema would have a node representation in the maximum skeleton, but not in the effective skeleton.

By comparing maximum and effective skeletons, we find advantages and disadvantages for both of them: The maximum skeleton is the more generalized skeleton version, and therefore it applies to many different network instances of the same RDF schema. However, its generality also means that it carries less specific information about each single instance, and therefore, conclusions drawn from a maximum skeleton are weaker than those drawn from an effective skeleton. The effective skeleton is specific to one instance of a semantic network. Thus, it cannot be reused for other instances, but it results in more precise conclusions.

C. Annotations

While the skeleton structure helps to understand the basic structure of the source network, a detailed analysis often requires more information: It might be useful to know, how many node or edges are subsumed by a node or edge representative in the skeleton; The average number of incoming or outgoing edges for all represented nodes could indicate a certain spreading behaviour; Maybe there are 10.000 edges of the same type subsumed by one edge representative, but actually they all originate in only 10 different nodes. To capture such (often numerical) information, skeletons can be enhanced by annotations. Typically, there are four types of annotations: those that describe node or edge representatives and those that describe the source or target of an edge representative.

Since effective and maximum skeletons carry different information, this also applies to annotations on them. While annotations on an effective skeleton refer to a concrete network instance of an RDF Schema (e.g., the concrete count of instances of a node type), annotations on a maximum skeleton describe potential values. Thus, an instance count could have the value *, meaning that any number of instances is possible.

D. Syntax

Let L_S be a set of labels. A Semantic Network Skeleton S is defined by

$$S = (N_S, E_S, s, t, l, \omega),$$

where

- N_S is a non-empty set of *node representatives*,
- E_S is a set of *edge representatives*,
- $s : E_S \rightarrow N_S$ is the *edge source mapping*,
- $t : E_S \rightarrow N_S$ is the *edge target mapping*, and
- $l : N_S \cup E_S \rightarrow L_S$ is the labelling,
- $\omega : E_S \rightarrow \mathbb{R}$ is the edge weight mapping.

The node and edge representatives each represent a set of nodes/edges of the same type from the original semantic network. Each edge representative $e \in E_S$ has a source node representative $s(e)$ and a target node representative $t(e)$. Furthermore, all node and edge representatives have a label $l(n)/l(e)$ assigned.

Given a semantic network skeleton $S = (N_S, E_S, s, t, l, \omega)$, and let $n_1, n_2 \in N_S$, $e \in E_S$, $s(e) = n_1$, and $t(e) = n_2$. Then the triple

$$T_S = (n_1, e, n_2)$$

is called a *skeleton triple* of S . A skeleton triple represents all corresponding RDF triples of the source network.

For a skeleton S the *skeleton annotation* A_S is defined as

$$A_S = (A_n, A_e, A_s, A_t),$$

where

- $A_n : K \times N_S \rightarrow V$ is the *node annotation*,
- $A_e : K \times E_S \rightarrow V$ is the *edge annotation*,
- $A_s : K \times E_S \rightarrow V$ is the *edge source annotation*, and
- $A_t : K \times E_S \rightarrow V$ is the *edge target annotation*.

Here, K stands for a set of *annotation keys*, and V stands for a set of *annotation values*.

E. Graphical Notation

The graphical notation for the skeleton corresponds to the graphical notation of RDF Graphs. In Figure 2, the proposed graphical notation is depicted. A node representative $n \in N$ is represented by a circle with its label $l(n)$ denoted over the circle. An edge representative $e \in E$ is represented by an unidirectional arrow with its label $l(e)$ denoted next to the arrow center. An arrow must connect two circles, with the arrow start connecting to the circle that represents the source and the tip of the arrow connecting to the circle that represents the target. Annotations are denoted in the circles, or near the start, middle, or end of the arrow, depending on their annotation type (node, edge, edge source, or edge target annotation).

F. Formal Notation of Graphical Example

A skeleton $S = (N_S, E_S, s, t, l, \omega)$ that contains among others the node and edge representatives depicted in Figure 2 would be formally denoted by

- the labels *Function*, *Malfunction*, *hasMalfunction* $\in L_S$,
- two nodes $n_1, n_2 \in N_S$ with $l(n_1) = \textit{Function}$, and $l(n_2) = \textit{Malfunction}$,
- an edge $e \in E_S$ with $l(e) = \textit{hasMalfunction}$, $s(e) = n_1$, and $t(e) = n_2$.

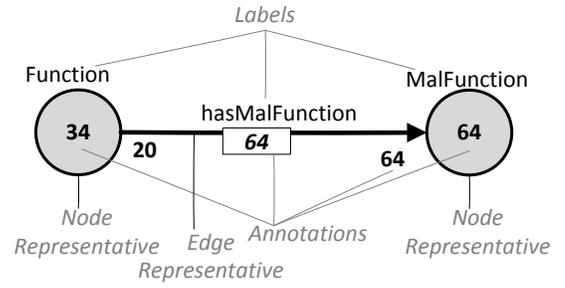


Figure 2. Graphical Notation for Skeletons.

Additionally, the skeleton annotation $A_S = (A_n, A_e, A_s, A_t)$ would contain the following mappings:

- $A_n(nc, n_1) = 34$,
- $A_n(nc, n_2) = 64$,
- $A_e(ec, e) = 64$,
- $A_s(src_rep, e) = 20$, and
- $A_t(tgt_rep, e) = 64$.

Here, nc and ec are the numbers of nodes/edges (node count and edge count) that have been subsumed by a node/edge representative. The source and target annotations src_rep and tgt_rep are the number of represented nodes that are part of represented RDF triples. Thus, 20 of the 34 nodes represented by n_1 are connected to nodes represented by n_2 via an edge represented by e . For the sake of brevity, we use the annotation keys similar as functions in the remainder of this article.

G. Skeleton Retrieval

Semantic network structures are as diverse as their potential applications and user-specific design decisions. Generally, skeletons can be retrieved from all kinds of semantic networks. However, transformation rules must guarantee that the semantic definition described in Section IV-A holds. We focus on retrieving skeletons from semantic networks based on RDF and RDF Schema. More specifically, we utilize the RDF statements from the corresponding RDF Graph. Technically, different approaches are possible from successively parsing RDF Statements to utilizing query languages such as SPARQL [19]. We offer an abstract retrieval description focusing on semantic compliance as introduced in [1].

H. Creating Effective Skeletons

For retrieving the effective node and edge representatives from the spread graph, we apply the following abstract method.

- 1) Each resource that is an RDF class becomes a node representative in the skeleton.
- 2) All instances of one class are subsumed by one node representative.
- 3) All literals are subsumed by one node representative in the skeleton.
- 4) For each statement, an edge representative is added (if not yet existent) for the predicate between the node representative of the statement's subject and the node representative of the statement's object in the skeleton.

Additionally, during the skeleton retrieval process, the desired annotation values can be computed. We propose to subsume all literals by one node representative in the skeleton. In RDF, the literals of the class `rdfs:Literal` contain literal values

such as strings and integers. A literal consists of a lexical form, which is a string with the content, a datatype IRI, and optionally a language tag. It is, of course, possible to further distinguish depending on datatype, or even analyzing value equality instead of term equality. However, the content string of the lexical form seems to be most important and sufficient for the application.

I. Creating Maximum Skeletons

For creating a maximum skeleton, we apply the following method to retrieve node and edge representatives from a spread graph.

- 1) Each resource that is an RDF class becomes a node representative in the skeleton. Additionally, a node representative for instances of this class must be created. For resources that are classes themselves and subclasses of another class all properties must be propagated from its superclass.
- 2) Find all properties and their scope (range, domain). For each property add (if not existent yet) an edge representative from the node representative for the instances of the specified domain to the node representative for the instances of the specified range. For each subproperty p_1 of a property p_2 edge representatives must be created between all node representatives connected via p_2 .

Again, required annotation values can be computed during the skeleton retrieval process.

V. SIMULATING SPREADING ACTIVATION BEHAVIOR WITH SEMANTIC NETWORK SKELETONS

Structural network properties as well as spreading activation constraints and configuration settings affect spreading activation results. Knowledge about spreading activation effects and their causes supports pre-configuration analyses in order to optimize the settings to retrieve the desired effects. Since semantic networks tend to be very large, an excessive examination of spreading behavior with a multitude of configuration settings can be a time-consuming task. Therefore, we utilize the comprised structural summary of semantic network skeletons to simulate the spreading activation behavior of its represented semantic network. In this section, we formally introduce the spreading activation simulation approach. For better comprehensibility, we apply selected simulation steps to an example.

A. Simulation Method

Spreading Activation on a network skeleton requires careful mapping of the algorithm to the new graph structure. Since the skeleton contains representatives for nodes and edges, we introduce an averaging approach in order to simulate the expected spreading behavior and approximate the activation strength and spreading strength for each simulation step.

In Figure 3, one challenge of this mapping can be observed. A spreading activation step on a skeleton triple needs to simulate spreading activation on the underlying bipartite graph. The explicit annotations are advantageous for the approach. We can identify how many represented nodes are not connected to a represented edge, and consequently can be identified as unreachable. Additionally, we can make estimations about the number of represented edges a represented node can be expected to be connected to. However, we do not have full

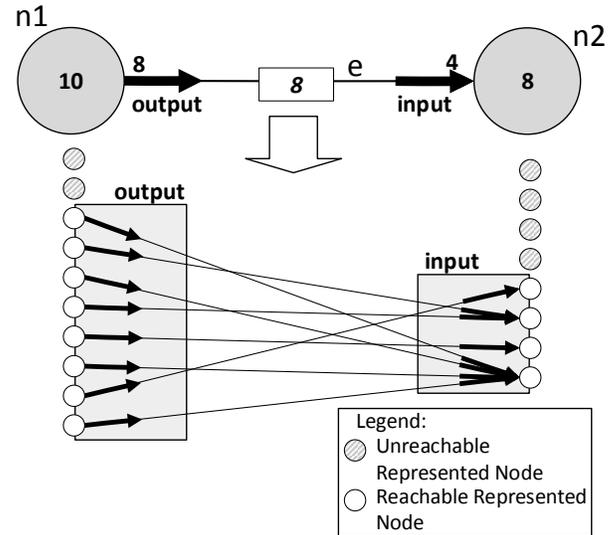


Figure 3. Spreading Activation on a Skeleton Triple.

information about the connectedness of the represented nodes and edges. Moreover, we have to pay attention to the actual state of expected activation in every pulse. For example, there is a difference for both input and output function if one node represented by n_1 is expected to be already activated, or if all of the 10 represented nodes are expected to be activated. We refer to the ratio of activated nodes in the set of represented nodes as saturation. The overlapping of already activated nodes and newly activated nodes needs to be considered as well, e.g., how many of the new ones are expected to be contained in the set of the already activated nodes. Therefore, we include combinatorial considerations into our spreading activation function mapping.

1) *Local And Global Simulation Steps:* We distinguish between two kinds of spreading steps that we refer to as local and global simulation steps. This differentiation supports the comprehensibility of the required adaptations. Figure 4 depicts local and global simulation areas.

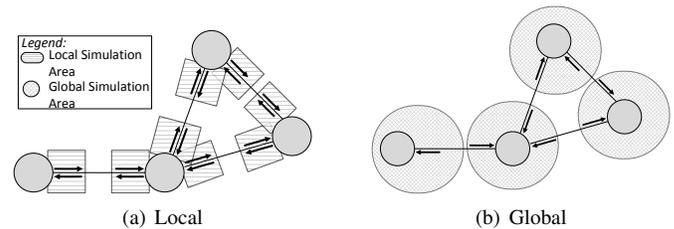


Figure 4. Simulation Areas.

In a local simulation step, each node-edge pair of the skeleton is examined. For each of these pairs, the expected outgoing as well as expected incoming activation are calculated.

Since a node representative may receive activation via various edge representatives, a consolidation of incoming activation is necessary. In a global simulation step, each node is examined and the results from the local observations are consolidated. Thus, we obtain the expected and incoming activation as well as the new expected overall activation level for each node in the skeleton.

2) *Simulation Input*: Spreading activation simulation requires the corresponding semantic network skeleton to the source network that the spreading is simulated for. We additionally need the same configuration settings, e.g., selected from constraints and spreading modes presented in Section III-B and III-C. Starting point for the simulation has to be the node representative(s) of the starting node(s) in the source network.

3) *Simulation Results*: Two aspects of activation distribution are of special interest. First, the expected growth of the number of activated nodes represented by each skeleton node. It reveals the *spreading strength* and answers the questions how fast activation probably reaches representatives of skeleton nodes in the underlying semantic network. More importantly, it examines the pulse-wise growth of the number of activated nodes represented by each skeleton node and provides an estimation about the proportion of already activated nodes represented by its representative, which we call the activation saturation. Second, we are interested in expected growth of the activation values of the nodes represented by a skeleton node. This reveals the *activation strength* and answers the question how fast activation values can be expected to increase in the underlying semantic network. Therefore, we calculate for each skeleton node the expected number of activated nodes and the expected activation value for each simulated spreading activation step.

B. Mapping Spreading Activation Functions for Simulation Purposes

The basic idea is to adapt the beforementioned three components of the spreading activation computation, i.e., output activation, input activation, and activation level, to an averaging approach for approximating the activation value development. Therefore, we introduce the following spreading activation simulation functions along with the corresponding levels of expected activation. For $n \in N_S$, edge $e \in E_S$, pulse $p \geq 0$, for local simulation steps:

- the aggregated output function $\overline{out} : N_S \times E_S \times \mathbb{R} \rightarrow \mathbb{R}$ determines the expected total output activation of node n via edge e , denoted by $\overline{o}_{n,e}^{(p)}$,
- the counting output function $out : N_S \times E_S \times \mathbb{R}^2 \rightarrow \mathbb{R}$ determines the expected number of nodes represented by n to be activated via edge e from node n , denoted by $\hat{o}_{n,e}^{(p)}$,
- the aggregated input function $\overline{in} : N_S \times E_S \times \mathbb{R} \rightarrow \mathbb{R}$ determines the expected total input activation of node n via edge e , denoted by $\overline{i}_{n,e}^{(p)}$,
- the counting input function $in : N_S \times E_S \times \mathbb{R}^3 \rightarrow \mathbb{R}$ determines the expected number of nodes represented by n to be newly activated via edge e , denoted by $\hat{i}_{n,e}^{(p)}$,

and for global simulation steps:

- the aggregated input function $\overline{in} : N_S \times N \rightarrow \mathbb{R}$ determines the expected total input activation of node n via all connected edges of node n , denoted by $\overline{i}_n^{(p)}$,
- the counting input function $\hat{in} : N_S \times N \rightarrow \mathbb{R}$ determines the expected number of nodes represented by n to be newly activated via all connected edges, denoted by $\hat{i}_n^{(p)}$,
- the aggregated activation function $\overline{act} : \mathbb{R}^2 \times \mathbb{R} \rightarrow \mathbb{R}$ determines the expected total activation of node n , denoted by $\overline{a}_n^{(p)}$,

- the counting activation function $\hat{act} : \mathbb{R}^2 \rightarrow \mathbb{R}$ determines the expected number of nodes represented by n to be newly activated, denoted by $\hat{a}_n^{(p)}$.

The skeleton triple and its underlying bipartite graph depicted in Figure 3 reveal two important facts. First, we can obtain the *connection rate* of a node represented by $n \in N_S$ to be connected with a represented edge of $e \in E_S$. This connection rate $con : N_S \times E_S \rightarrow \mathbb{R}$ can be calculated as follows:

$$con(n, e) := \begin{cases} \frac{src_rep(e)}{nc(n)} & \text{if } s(e) = n, \\ \frac{tgt_rep(e)}{nc(n)} & \text{if } t(e) = n, s(e) \neq n, \\ 0 & \text{else.} \end{cases} \quad (18)$$

We want to point out that loops are handled by the first case and restricted from the second case such that only one *con*-value per node and connected edge exists, i.e., in the reading direction of the edge's property.

Second, the expected spread factor $fac : N_S \times E_S \rightarrow \mathbb{R}$ denotes the number of edges represented by $e \in E_S$ that each connected node represented by $n \in N_S$ is expected to be connected to.

$$fac(n, e) := \begin{cases} \frac{ec(e)}{src_rep(e)} & \text{if } s(e) = n, \\ \frac{ec(e)}{tgt_rep(e)} & \text{if } t(e) = n \wedge s(e) \neq n, \\ 0 & \text{else.} \end{cases} \quad (19)$$

Figure 5 depicts an annotated skeleton triple with two node-edge pairs, each for a local simulation step in the output and in the input direction. This will be the example for the following simulation step descriptions.

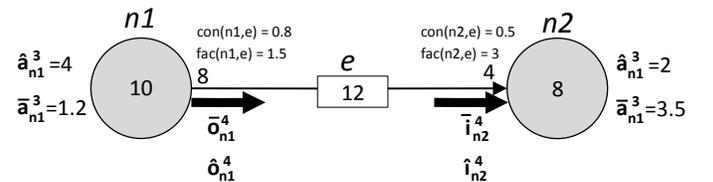


Figure 5. Annotated Skeleton Triple with expected activation levels assigned.

1) *Local Calculation of Expected Total Activation*: The expected total output activation can be calculated by an adapted constraints- and mode-aware output function. The spreading modes presented in this article and the associated permission function φ do not require adaptations when used for spreading activation simulation, as well as pulse decay and activation threshold. The fan-out constraint requires adapting the fan-out factor, since spreading in the skeleton follows an averaging approach. Here, we do not consider the degree of a node to be of interest but the expected spread factor of all edges connected to a node, defined by $fac : N_S \rightarrow \mathbb{R}$. We define $\overline{o}_{n_1,e}$ to be the expected total activation that might be passed from the nodes represented by n_1 via the connected edges represented by e . Therefore, we need to consider the connectivity rate as well as the spread factor in the aggregated output function. The total output activation value is determined by:

$$\overline{o}_{n,e}^{(p)} = \overline{out}(n, e, \overline{a}_n^{(p-1)}), \quad (20)$$

where

$$\overline{out}(n, e, a) := \begin{cases} d \cdot a \cdot \text{con}(n, e) \cdot \text{fac}(n, e) & \text{if } s(e) = n \vee t(e) = n, \\ & a \geq \tau, \neg \text{fanout}, \\ \frac{d \cdot a \cdot \text{con}(n, e) \cdot \text{fac}(n, e)}{\text{fac}(n)} & \text{if } s(e) = n \vee t(e) = n, \\ & a \geq \tau, \text{fanout}, \\ 0 & \text{else,} \end{cases} \quad (21)$$

where

$$\overline{fac}(n) := \sum_{e \in E_S, s(e)=n \vee t(e)=n} \text{fac}(n, e). \quad (22)$$

The expected total input activation $\overline{i}_{n,e}^{(p)}$ for node $n, m \in N_S$ via edge $e \in E_S$ at each pulse $p > 0$ is determined by the input function without further adaptations:

$$\overline{i}_{n,e}^{(p)} = \overline{in}(n, e, \overline{o}_m^{(p)}) \quad (23)$$

where

$$\overline{in}(n, e, o) := \begin{cases} o \cdot \omega(e) & \text{if } t(e) = n, \\ o \cdot \omega(e) & \text{if } s(e) = n, \\ 0 & \text{else.} \end{cases} \quad (24)$$

The example in Figure 6 shows that without any constraints the assigned expected activation value of $n1$ is expected to be passed by 8 nodes represented by $n1$ where each is expected to be connected to 1.5 of the represented edges. Therefore, the total expected output activation can be expected to be 1.44. Since there is no edge weights assigned, in this example the total expected input activation can be expected to be the total output activation 1.44.

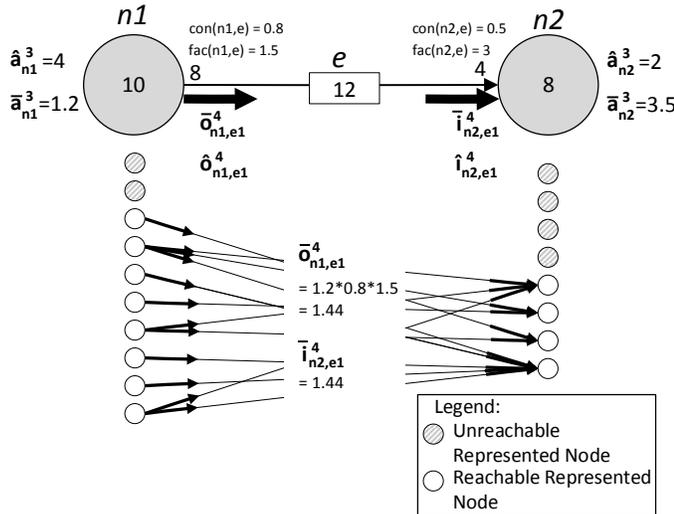


Figure 6. Example: Local Expected Total Activation.

2) *Local Calculation of Expected Number of Activated Nodes*: The expected number of activated output nodes can be calculated by an adapted mode- (and constraint) aware output function:

$$\hat{o}_{n,e}^{(p)} = \hat{out}(n, e, \hat{a}_n^{(p-1)}, \overline{o}_{n,e}^{(p)}), \quad (25)$$

where

$$\hat{out}(n, e, a, o) := \begin{cases} a \cdot \text{con}(n, e) \cdot \text{fac}(n, e) & \text{if } s(e) = n \vee t(e) = n, \\ & o > 0, \\ 0 & \text{else.} \end{cases} \quad (26)$$

In contrast to the computation of expected total activation, the local expected number of activated nodes does not directly depend on the spreading activation constraints. Indirectly, the restriction in the first case requires an expected total output activation greater than zero to be transported via edge e , and input activation respectively (which indirectly may be influenced by the algorithm settings).

The expected input activation number can be calculated by an adapted and constraint aware input function.

$$\hat{i}_{n,e}^{(p)} = \hat{in}(n, e, \hat{a}_n^{(p-1)}, \hat{o}_{m,e}^{(p)}, \overline{i}_{n,e}^{(p)}) \quad (27)$$

The input function needs to take into account that we have no knowledge about the actual connections between represented nodes and edges. When a certain number of edges reaches nodes, there are several potential combinations. In order to estimate how many represented nodes are reached by a number of activation passing edges, we follow a combinatorial approach. The function *split* determines how the number activation passing edges needs to be reduced regarding their receiving nodes. It represents a combinatorial approach to get the average number of buckets (y) that have a ball, when distributing x balls to them.

$$\text{split}(x, y) := \frac{1 - \left(\frac{y-1}{y}\right)^x}{1 - \left(\frac{y-1}{y}\right)} \quad (28)$$

$$\hat{i}_{n,e}^{(p)} = \hat{in}(n, e, a, o, i) := \begin{cases} \text{split}(o, \text{src_rep}(e)) \cdot \left(1 - \frac{a}{nc(n)}\right) & \text{if } s(e) = n, i > 0, \\ \text{split}(o, \text{tgt_rep}(e)) \cdot \left(1 - \frac{a}{nc(n)}\right) & \text{if } t(e) = n, i > 0, \\ 0 & \text{else} \end{cases} \quad (29)$$

Figure 7 depicts the example calculation for this simulation step. Only already activated and connected nodes can spread via the expected number of connected edges per node. After split reduction, the remaining nodes need to be checked for potential overlapping with already activated nodes. As a result, we can expect 2.25 nodes to be newly activated after this local simulation steps.

C. Global Simulation Step - Consolidation of Local simulation Steps

After examining each node-edge pair in the local simulation step, we must consolidate them for each node because one node may receive input via many edges.

1) *Global Calculation of Expected Total Activation*: The complete expected input activation value of a node n via all edges can be determined by the adapted input activation function:

$$\overline{i}_n^{(p)} = \overline{in}(n, p), \quad (30)$$

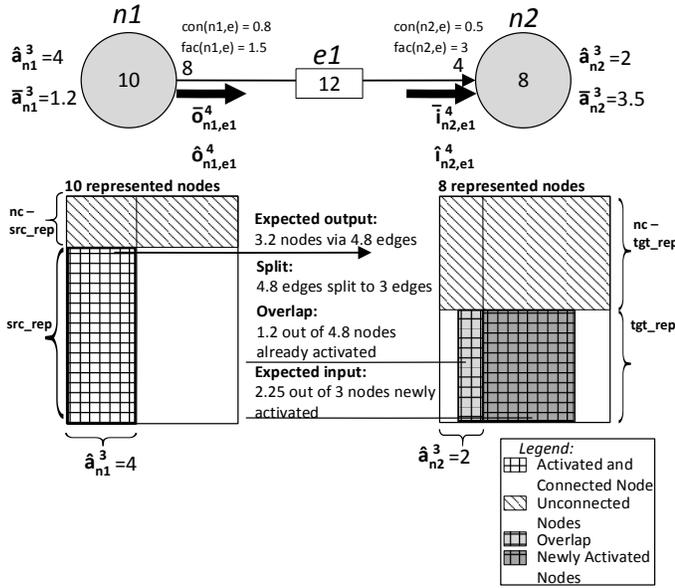


Figure 7. Example: Local Expected Number of Activated Nodes.

where

$$\bar{i}_n(n, p) := \sum_{e \in E_S} \bar{i}_{n,e}^{(p)}. \quad (31)$$

The complete expected activation level value of a node n via all edges can be performed by the activation function, no further adaptations needed:

$$\bar{a}_n^{(p)} = act(\bar{i}_n^{(p)}, \bar{a}_n^{(p-1)}). \quad (32)$$

Figure 8 depicts an example calculation. All expected input activation values for a node are aggregated and then used for the calculation of its new expected total activation.

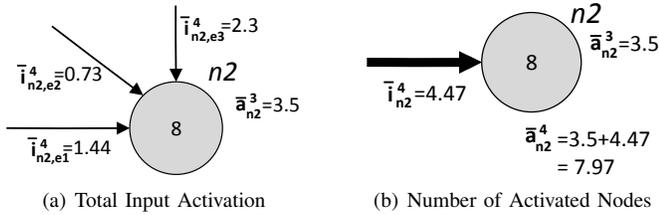


Figure 8. Example: Global Expected Activation.

2) *Global Calculation of Expected Number of Activated Nodes*: When consolidating, we have to take into account that the expected number of activated nodes transported via all connected edges to a node might contain overlap. First, a represented node that gets activated may already be contained in the set of activated nodes of the target node representative. Second, potentially newly activated nodes coming from different edges may have an overlap as well. Therefore, the consolidated expected number of potentially newly activated nodes should not contain the nodes that are expected to be not activated.

The number of non-activated represented nodes $non_act : N_S \rightarrow \mathbb{R}$ of a node representative n in a pulse p is defined as

$$non_act(n, p) := nc(n) - \hat{a}_n^{(p-1)}. \quad (33)$$

The number of represented nodes that are expected to be neither activated nor part of any overlap is denoted by $idle : N_S \rightarrow \mathbb{R}$:

$$idle(n, p) := non_act(n, p) \cdot \prod_{e \in E_S} \left(1 - \frac{\hat{i}_{n,e}^{(p)}}{non_act(n, p)}\right). \quad (34)$$

The expected input activation number via all edges is denoted by

$$\hat{i}_n^{(p)} = \hat{i}_n(n, p), \quad (35)$$

where

$$\hat{i}_n(n, p) := non_act(n, p) - idle(n, p). \quad (36)$$

Computation of the complete expected number of activated represented nodes of a node n can be performed by the activation function, no further adaptations needed.

$$\hat{a}_n^{(p)} = act(\hat{i}_n^{(p)}, \hat{a}_n^{(p-1)}) \quad (37)$$

Figure 9 depicts an example calculation. All expected input activation values for a node are aggregated and then used for the calculation of its new expected total activation.

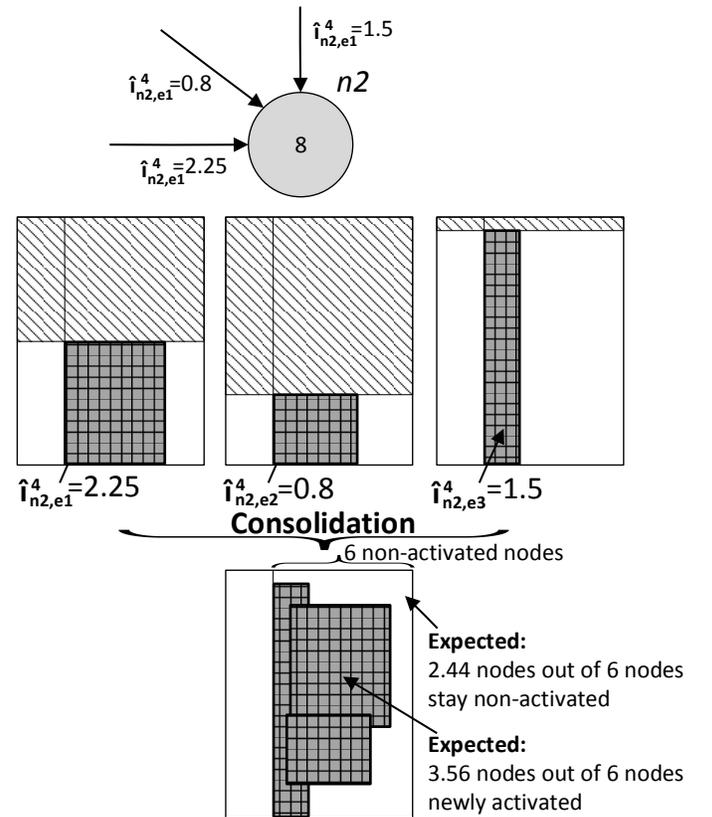


Figure 9. Example: Global Expected Number of Newly Activated Nodes.

VI. EXPERIMENTS

A. Appropriateness of Simulation Results

Predictive precision is a key property of a good simulation. Similarly, this should apply for spreading simulation on a semantic skeleton. To understand the quality of the predictions made by spreading simulation, we will compare pulses of a

simulation on skeleton triples with real spreading pulses on the underlying network.

When simulating a pulse, we are mainly interested in a good prediction for the number of activated nodes and for the total activation of the receiving node representative. However, there are many other values that can be predicted by the simulation as well, such as the number of edges that transported activation from source to target and how much activation has been transported by them.

To compare simulation and spreading results, we generated 5000 pairs of skeleton triples $T_S = (n_1, e, n_2)$ with their corresponding underlying networks, and used each pair with the same activation values. To observe a high number of diverse cases, we decided for rather large triples with $A_n(nc, n_1) = 100$, $A_n(nc, n_2) = 130$, and $A_e(ec, e)$ being randomly generated, averaging at around 200. We kept the number of edges low in order to guarantee examples with connected and unconnected nodes, and therefore, to avoid trivial examples. On average, we assigned initial activation between 0.0 and 10.0 to 40% of the represented nodes of n_1 . In order to observe the local consolidation, we assigned initial activation 0.0 and 10.0 to 20% of the represented nodes of n_2 .

For our initial experiment, we decided to use pure spreading, i.e., a threshold of 0.0, a decay of 1.0, and no fanout. We collected the mean and standard deviation of the spreading results, the simulation results, and of the difference between spreading result and simulation result in Table I. Additionally, we calculated the difference mean as well as the difference standard deviation relative to the spreading result (shown as % diff).

Table I shows that total activation values can be predicted with high accuracy. The mean prediction error is far below 1% of the spreading average for total edge activation as well as right total activation. The standard deviation of the error is also very acceptable for simulating pulses. Predicting activated node counts is less precise, with up to -7% mean error. Nevertheless, this value improved at the end of the pulse to -4.7% . Altogether, the simulation shows a very good estimate of the total activation of n_2 after the pulse, and a good estimate of the number of activated nodes.

After the base experiment, we examine some variations in order to see if they affect the accuracy of the predictions. In the second experiment (see Table II), we activated the fanout constraint. Thus, nodes will spread less activation, depending on their connectivity. As we can see, the simulation also predicts this fanout parameter well. There is even an improvement on the standard deviation of the error, which is half the deviation of the base example.

In the third experiment (see Table III), we activated the decay factor and set it to 0.5. Thus, again, nodes will spread less activation, but this time independent of other factors. Again, the simulation predicts this parameter well, but there is no significant change compared to the base experiment.

In Experiment 4 (see Table IV), we set the activation threshold to 3.0. So, this times less nodes will spread activation. The experiment indicated that thresholds are a bit harder to predict, with a mean error of -0.3% . Still, this value is a very good prediction.

Experiment 5 (see Table V) combines the three parameters of experiments 2-4. The mean error didn't change too much, but interestingly, the standard deviation of the error dropped further to 2.6%

TABLE I. Experiment 1: Base

| total edge activation $\bar{o}_{n_1}^p$ | mean | standard deviation |
|--|-------------|---------------------------|
| spreading | 399.696 | 78.992 |
| simulation | 400.078 | 66.428 |
| diff | 0.382 | 42.684 |
| % diff | 0.095% | 10.679% |
| active edge count $\hat{o}_{n_1}^p$ | | |
| spreading | 79.807 | 13.084 |
| simulation | 79.891 | 11.160 |
| diff | 0.083 | 6.753 |
| % diff | 0.105% | 8.462% |
| right receiving node count $split(\hat{o}_{n_1}^p, tgt_rep(e))$ | | |
| spreading | 59.758 | 7.724 |
| simulation | 55.389 | 5.460 |
| diff | -4.369 | 4.659 |
| % diff | -7.311% | 7.797% |
| right new active node count $\hat{i}_{n_2}^p$ | | |
| spreading | 47.804 | 6.899 |
| simulation | 44.316 | 4.810 |
| diff | -3.488 | 4.305 |
| % diff | -7.297% | 9.006% |
| right total activation $\bar{a}_{n_2}^p$ | | |
| spreading | 529.567 | 83.178 |
| simulation | 529.949 | 71.584 |
| diff | 0.382 | 42.684 |
| % diff | 0.072% | 8.060% |
| right active node count $\hat{a}_{n_2}^p$ | | |
| spreading | 73.801 | 6.935 |
| simulation | 70.313 | 5.045 |
| abs. diff | -3.488 | 4.305 |
| % diff | -4.727% | 5.833% |

TABLE II. Experiment 2: Fanout

| right total activation $\bar{a}_{n_2}^p$ | mean | standard deviation |
|---|-------------|---------------------------|
| spreading | 303.203 | 39.914 |
| simulation | 303.273 | 38.196 |
| diff | 0.070 | 10.445 |
| % diff | 0.023% | 3.445% |
| right active node count $\hat{a}_{n_2}^p$ | | |
| spreading | 73.912 | 7.034 |
| simulation | 70.356 | 5.092 |
| abs. diff | -3.557 | 4.402 |
| % diff | -4.812% | 5.956% |

TABLE III. Experiment 3: Decay

| right total activation $\bar{a}_{n_2}^p$ | mean | standard deviation |
|---|-------------|---------------------------|
| spreading | 330.468 | 48.010 |
| simulation | 330.159 | 43.124 |
| diff | -0.309 | 21.607 |
| % diff | -0.094% | 6.538% |
| right active node count $\hat{a}_{n_2}^p$ | | |
| spreading | 74.037 | 7.152 |
| simulation | 70.365 | 5.146 |
| abs. diff | -3.672 | 4.429 |
| % diff | -4.959% | 5.983% |

TABLE IV. Experiment 4: Threshold

| right total activation $\bar{a}_{n_2}^p$ | mean | standard deviation |
|---|-------------|---------------------------|
| spreading | 495.542 | 83.702 |
| simulation | 494.037 | 70.426 |
| diff | -1.504 | 44.059 |
| % diff | -0.304% | 8.891% |
| right active node count $\hat{a}_{n_2}^p$ | | |
| spreading | 62.666 | 7.351 |
| simulation | 60.187 | 5.038 |
| abs. diff | -2.479 | 4.903 |
| % diff | -3.956% | 7.824% |

TABLE V. Experiment 5: All Parameters

| right total activation \hat{a}_{n2}^p | mean | standard deviation |
|--|---------|--------------------|
| spreading | 209.716 | 31.117 |
| simulation | 209.390 | 30.425 |
| diff | -0.327 | 5.580 |
| % diff | -0.156% | 2.661% |
| right active node count \hat{a}_{n2}^p | | |
| spreading | 62.527 | 7.362 |
| simulation | 60.152 | 5.101 |
| abs. diff | -2.375 | 4.895 |
| % diff | -3.798% | 7.828% |

Altogether, the results look promising, and it seems like using all parameters pushes the predictive power of the simulation.

B. Time-Related Advantages

Besides increased comprehensibility, another main motivation for the concept of semantic skeletons and spreading simulation is the expected time gain for potential applications. Of course, we do not aim at replacing spreading activation but propose bypassing potential applications whenever the averaging simulation results are sufficient. Therefore, we examined the run time of both spreading activation on a semantic network and spreading simulation on its skeleton.

1) *Experimental Setup*: The semantic network under investigation is taken from our related research on an advisory system for decision-making support for hazard and risk analysis in the automotive domain [20]. This analysis is strongly expert-driven, and therefore expensive, time consuming, and dependent from the individual experts opinion. Therefore, the advisory system aims at providing useful recommendations for expert users when conducting such safety analyses. The semantic search within this network is performed by spreading activation. This advisory system will be utilized for our time-related experiments.

The network contains the knowledge taken from more than 150 concluded safety analyses from the domain. It contains more than 257.000 edges (representing 45 properties) connecting more than 61.800. The retrieved effective network skeleton only consists of 136 skeleton triple containing 45 node representatives and 136 edge representatives.

For the experiments, we examined 36 input variants using combinations of the different constraints and spreading modes presented in this article. Each input variant was executed for two different starting node situations. The experiments are conducted in the same running environment for both the spreading and the simulation approach.

2) *Results*: Figure 10 depicts the elapsed average time for the spreading activation runs as well as for the simulation runs. The simulation's lead in performance is obvious. The graph shows a gradual increase of run time over the pulses for the simulation runs. In contrast, the spreading activation graph reveals a sharp rise in execution time. One reason, why simulation can be performed faster is the smaller network size of a skeleton. We also observe differences regarding their configurations. Since both the simulation and the spreading runs are performed with the same input, we assume that it does not favor any of the both.

The results support our motivation to bypass the difficulties caused by extensive semantic network sizes by taking advantage of the skeleton's compactness. We conclude that whenever the averaging results from the simulation approach

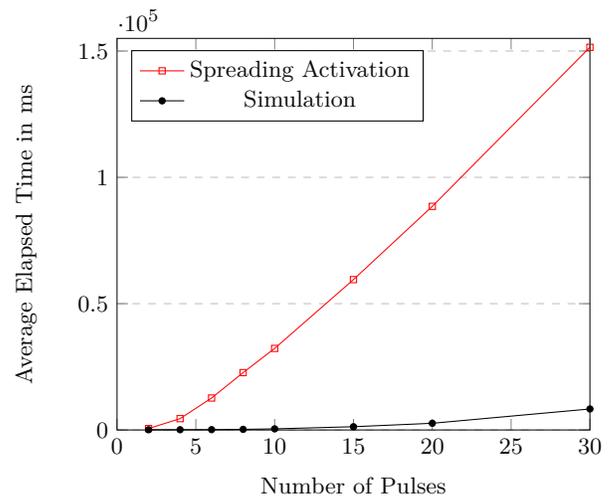


Figure 10. Average Elapsed Time for Spreading Activation and Spreading Simulation Runs.

are sufficient for a specific application or analysis, e.g., effect detection in pre-configuration analyses, the use of skeletons can save time.

VII. CONCLUSION AND FUTURE WORK

In this article, we extended previous work on the concept of semantic network skeletons by an approach that utilizes skeletons for simulating spreading activation. We presented a framework for spreading activation simulation that supports detailed observations of two basic properties: the expected pulsewise development of activation values, and the number of nodes that are expected to be activated, which is a measure for activation saturation in the semantic network. The approach is based on a careful mapping of the spreading activation functions to the specific characteristics of a skeleton. Averaging and combinatorial methods are used in order to estimate the spreading and activation behavior for the represented nodes and edges.

Through randomized experiments, we showed that the simulation results are good predictors for the actual spreading activation on the original network. We furthermore showed in time-related experiments that spreading simulation on a skeleton outperforms spreading activation on its represented semantic network.

We claimed that proper configuration is crucial and requires special attention for the useful application of spreading activation as a semantic search method in most application areas. Simulation results can reveal valuable information about spreading and activation behavior, e.g., the effects that may occur under certain influences from the algorithm and network settings. We showed that simulation can be performed on the comprised structure of a skeleton in a more efficient and sufficiently approximated way. Therefore, we conclude that it enables more efficient pre-configuration analyses.

The skeleton is not a tool to avoid spreading activation on semantic networks, but it poses an alternative to enhance working with vast networks that are difficult to manage. The objective is to use a skeleton's capabilities, i.e., time advantage and comprehensibility, to improve processes in the background in order to make spreading activation activities in the foreground easier to use with better results.

The semantic skeleton offers many opportunities for further applications and advanced extensions. Follow-up research can set up catalogs of potential structural and configurational influences as well as their effects in a systematic way in order to identify correlations and interdependencies. The resulting guidelines can be of direct use for finding proper configurations by including desired effects or excluding undesired effects. This may replace the momentary experience-driven controlling of such algorithms and may support a consistent rise in the quality of semantic search algorithms. Applicability and usability of spreading activation approaches may increase since manual adaptations often pose obstacles.

However, we have the futuristic vision of an adaptive self-configuration approach that automatically determines alleged good or bad configurations. For this long term goal, networks that change in size and structure should be treated flexibly such that disadvantages coming from hard-coding algorithm parameters can be discarded.

Skeletons can be valuable beyond utilization in the context of spreading activation. The compressed and summarized character of network skeletons might be useful for cognate disciplines also dealing with large semantic networks.

REFERENCES

- [1] K. Hartig and T. Karbe, "Semantic Network Skeletons - A Tool to Analyze Spreading Activation Effects," in The Eighth International Conference on Information, Process, and Knowledge Management eKNOW 2016, C. Granja, R. Oberhauser, L. Stanchev, and D. Malzahn, Eds., 2016, pp. 126–131.
- [2] F. Crestani, "Application of Spreading Activation Techniques in Information Retrieval," *Artificial Intelligence Review*, vol. 11, no. 6, 1997, pp. 453–482.
- [3] M. R. Quillian, "Semantic Memory," in *Semantic Information Processing*, MIT Press, Ed., 1968, pp. 216–270.
- [4] A. M. Collins and E. F. Loftus, "A Spreading-Activation Theory of Semantic Processing," *Psychological Review*, vol. 82, no. 6, 1975, pp. 407–428.
- [5] J. F. Sowa, *Principles of Semantic Networks: Exploration in the Representation of Knowledge*, J. F. Sowa and A. Borgida, Eds. Morgan Kaufmann, Jan. 1991.
- [6] "RDF 1.1 Concepts and Abstract Syntax," W3C, W3C Recommendation, Feb. 2014.
- [7] J. R. Anderson, "A Spreading Activation Theory of Memory," *Journal of Verbal Learning and Verbal Behavior*, 1983, pp. 261–295.
- [8] M. R. Berthold, U. Brandes, T. Kötter, M. Mader, U. Nagel, and K. Thiel, "Pure Spreading Activation is Pointless," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management CIKM 2009*, New York, NY, USA, 2009, pp. 1915–1918.
- [9] J. M. Álvarez, L. Polo, and J. E. Labra, "ONTOSPREAD: A Framework for Supporting the Activation of Concepts in Graph-Based Structures through the Spreading Activation Technique," *Information Systems, E-learning, and Knowledge Management Research*, vol. 278, 2013, pp. 454–459.
- [10] J. M. Alvarez, L. Polo, W. Jimenez, P. Abella, and J. E. Labra, "Application of the spreading activation technique for recommending concepts of well-known ontologies in medical systems," in *Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine - BCB 2011*, R. Grossman, A. Rzhetsky, S. Kim, and W. Wang, Eds. New York, New York, USA: ACM Press, 2011, pp. 626–635.
- [11] L. Grad-Gyenge, H. Werthner, and P. Filzmoser, "Knowledge Graph based Recommendation Techniques for Email Remarketing," *International Journal on Advances in Intelligent Systems*, vol. 9, no. 3&4, 2016.
- [12] F. Crestani and P. L. Lee, "Searching the web by constrained spreading activation," *Information Processing & Management*, vol. 36, no. 4, 2000, pp. 585–605.
- [13] C.-N. Ziegler and G. Lausen, "Spreading Activation Models for Trust Propagation," in *IEEE International Conference on e-Technology, e-Commerce, and e-Services EEE 2004*. Los Alamitos and Piscataway: IEEE Computer Society Press, 2004, pp. 83–97.
- [14] K. Schumacher, M. Sintek, and L. Sauermaun, "Combining Fact and Document Retrieval with Spreading Activation for Semantic Desktop Search," in *Proceedings of the 5th European Semantic Web Conference*, ser. *Lecture Notes in Computer Science*, S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis, Eds. Springer, 2008, vol. 5021, pp. 569–583.
- [15] J. Banks, Ed., *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. New York, NY, USA: Wiley, 1998.
- [16] J. A. Sokolowski and C. M. Banks, Eds., *Principles of Modeling and Simulation: A Multidisciplinary Approach*. Hoboken N.J.: John Wiley, 2009.
- [17] M. D. Haselman, S. Hauck, T. K. Lewellen, and R. S. Miyaoka, "Simulation of Algorithms for Pulse Timing in FPGAs," *IEEE Nuclear Science Symposium conference record. Nuclear Science Symposium*, vol. 4, 2007, pp. 3161–3165.
- [18] J. Polvichai, P. Scerri, and M. Lewis, "An Approach to Online Optimization of Heuristic Coordination Algorithms," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems AAMAS 2008 - Volume 2*, L. Padgham, D. C. Parkes, J. Mueller, and S. Parsons, Eds. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 623–630.
- [19] "SPARQL 1.1 Query Language," W3C, W3C Recommendation, Mar. 2013.
- [20] K. Hartig and T. Karbe, "Recommendation-Based Decision Support for Hazard Analysis and Risk Assessment," in *The Eighth International Conference on Information, Process, and Knowledge Management eKNOW 2016*, C. Granja, R. Oberhauser, L. Stanchev, and D. Malzahn, Eds., 2016.