Interactive Visualization of Continuous Node Features in Graphs

Stefan Haun, Marcus Nitsche, Andreas Nürnberger

Data and Knowledge Engineering Group,
Departement of Computer Science,
Institute of Technical and Business Information Systems,
Otto-von-Guericke-University of Magdeburg,
Universitätsplatz 2, 39106 Magdeburg, Germany

{stefan.haun, marcus.nitsche, andreas.nuernberger}@ovgu.de

Abstract: Ordinary graphs only support discrete structures. In this paper we present an approach towards embedding continuous data – like time stamps or series of measurements – in discrete graph models. These continuous meta-information implicitly define relations between vertices which are not explicitly defined in the graph itself. We call this an induced Non-Discrete Graph Structure (NoDeS). The model is helpful for visualization of time-dependent models or values from physical domains. We provide a formal definition of NoDeS based on graphs and two mappings, instance and annotation based, to already known graph structures and visualizations. A visualization of multi-partite projection provides a representation of information from several contexts, enabling NoDeS for a generic context switching mechanism which is used for interaction with these structures. Finally, we introduce an application concept for agent-driven event scheduling using NoDeS.

1 Introduction

In discrete mathematics graphs are a solid theoretical approach to model and visualize relations between items and to visualize complex structures in many different areas (cf. [DL07]). While these structures are static and discrete we thought about adding continuous data to classic graph structures. This supports visualizing relations between instances as well as sticking continuous node features like time measurements for instance. In this paper we present the formal description of this enhanced graph structure as well as interaction sequences of end-users.

We first give a brief overview of related work in the area of graph visualization and systems related to web searching and the visualization of hypermedia structures. Then we introduce our approach towards embedding continuous domains into graphs. A concept for the interactive visualization of the presented structure is discussed in Section 4. Finally, in Section 5, we present an application example based on the problem of Agent-driven event scheduling to show how our approach can be used.

2 Related Work

Related work can be found in the field of graph visualization and graph layouting. Cook and Holder, although mostly concerned with graph mining, provide a good overview on the state of the art and current systems, [CH07]. For a general overview there are several surveys on graph visualization available (c.f. [HMM00], [DBETT94], [Tam99]). According to [DL07], there are three major methods for graph layouting: *force-directed*, *hierarchical* and *topology-shape-metrics*, where the force directed method introduced by [Ead84] is most used today, despite its disadvantageous behavior in interactive systems [HMM00], such as presented in Sec. 5. Special visualizations can be used to accommodate data specific features such as time lines: [DRS04] introduces a 2.5D time-series data visualization, which uses stacks to represent time-dependent advances in data series. However, all existing layout and visualization methods do not take continuous domains into account and rather rely on the inherent relations of the continuous data, such as ordering and distance, being encoded into the discrete graph structure.

A large number of visualization systems is available. Approaches tailored to web searching and the visualization of hypermedia structures can be found among the web meta-search clustering engines (Vivísmo[1], iBoogie[2], SnakeT[3], WhatsOnWeb[4]) and in the field of semantic wikis (iMapping Wiki [HKV06]).

Combining visualization methods become more interesting in the context of heterogeneous information networks as they are heavily used throughout the BISON project[5].

3 A Structure for Visualizing Continuous Node Features

For our approach, we use the graph definition as proposed in [Die06]:

> A *graph* is a pair $G = (V, E)$ of disjoint sets with $E \subseteq V^2$, i.e. the elements of E are two-element subsets of V, where elements from V are *vertices* and elements from E are *edges* connecting the vertices.

Often the term *node* is used when referring to vertices and edges may be called *links*, especially in hyperlink structures.

[1] http://vivismo.com/
[2] http://www.iboogie.com/
[3] http://snaket.di.unipi.it
[4] http://whatsonweb.diei.unipg.it
[5] http://www.bisonet.eu

We call a vertex v continuous if meta-data from a continuous domain is mapped to v. There are two different mappings to ordinary graphs:

Mapping with meta-data. In the first mapping, continuous information is attached as meta-data to existing vertices (cf. Fig. 1). If there is a complete linkage between a continuous and a discrete domain, i.e. for each vertex from the continuous domain there is a matching vertex from the discrete domain, the continuous information can be attached to the discrete vertices. This resembles ordinary graphs with embedded meta-information. However, since the continuous vertices are only attached to other vertices, they are not a genuine part of the graph structure. It is not possible to distinguish between the continuous vertex and the mapping vertex in the graph structure and it is not possible to label the edges as they do not explicitly exist. Also, algorithms need to take this mapping into account and have to recognize attached meta-information as distinct vertices.

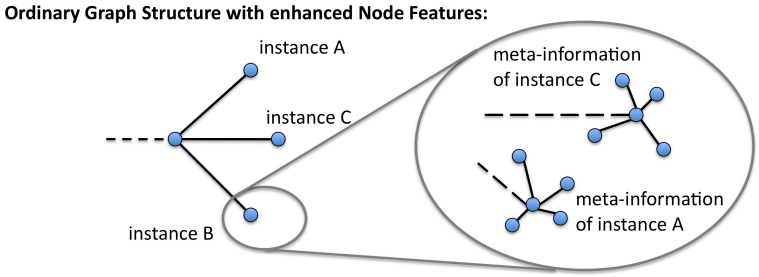


Figure 1: Graph with attached meta-information.

Mapping with meta-nodes. The second – preferred – mapping uses discrete nodes as instances from the continuous domain (cf. Fig. 2). Instead of attaching meta-information to existing nodes, we create meta-nodes which serve as a discrete instance of a value from the continuous domain. In contrast to the first mapping, these nodes do not represent any information other than the value from the continuous domain and yet can be treated as ordinary graph elements, i.e. meta-information can be attached and labeled edges can be added. However, two nodes representing the same value must be considered equal, i.e. before adding a meta-node, the graph must be searched for already existing nodes representing the desired value. As an advantage of this approach the resulting graph structure may contain continuous domains and still is an ordinary graph where existing algorithms can be applied. As a disadvantage meta-nodes do not represent relationships between elements from the continuous domain, i.e. two distinct nodes may have a very large distance, but also represent two very similar values. A distinction function needs to be used to decide whether two nodes should be considered equal to avoid cluttering the graph with similar value representations (with the penalty of imprecise representation of the continuous domain), which can introduce model knowledge about the underlying data, e.g. by using fuzzy sets.

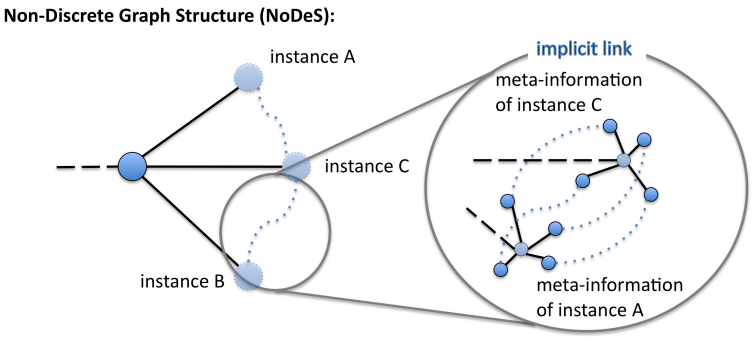


Figure 2: Interconnected meta-information, visualized as implicit links between vertices.

While both approaches result in an ordinary, annotated graph, the continuous graph structure differs in its semantics. The continuous vertices are bridges into the continuous domain rather than independent discrete elements. There may be an indirect relationship between those continuous vertices, if it is implied by the continuous domain, e.g. some distance measure. When calculating distances in the graph, those measures may be taken into account.

To introduce continuous vertices, the set V is built by the union of r multiple, arbitrary domains

$$V = V_1 \cup V_2 \cup \ldots \cup V_r$$

where each domain $V_1, \ldots, V_r$ may be either discrete or continuous, i.e. values from a continuous domain are embedded by adding a vertex as its discrete representative. The resulting graph not only contains vertices from discrete domains, but also elements from the continuous domain and their implicit connections, which are formed by the intrinsic ordering and distance function of the respective domain. For example, a pair of numbers in $\mathbb{R}$ is implicitly linked by the distance between the numbers (their difference) and has an ordering which implies a directed edge in the graph. The set of edges in the non-discrete graph structure is built from the set of explicit edges and the set of all edges induced by the continuous domain:

$$E = E_{Discrete} \cup E_{Implicit}$$

This type of graph we call **No**n-**D**iscret**e** graph **S**tructure (NoDeS) induced by a domain.

4 Interactive Visualization of NoDeS

Interactive variants of a NoDeS visualization can be easily implemented by setting single nodes or sets of nodes sensitive for user interactions like *selecting*, *dragging* and *dropping*. With slight adaptations, the NoDeS structure is not only interactive, but also allows to act context sensitive in a users perspective. Therefore NoDeS provide a generic context switching concept which allows to transfer the graph structure to different types of applications. Discrete vertices and meta-vertices, representing continuous data, are linked to each other (cf. Fig. 3).

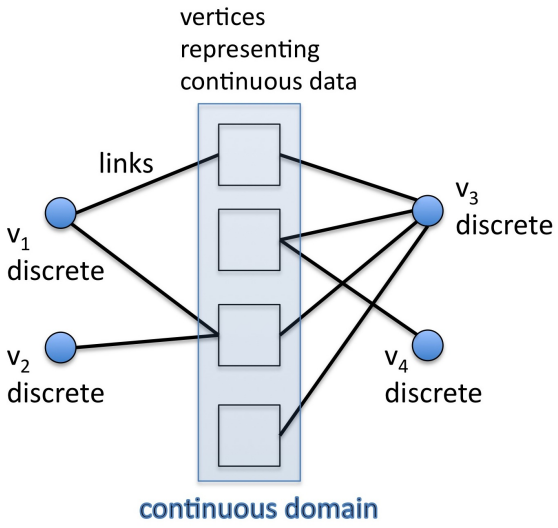


Figure 3: Generic structure of *NoDeS*.

Besides the indirect connections within the continuous domains, NoDeS can be defined as multi-partite graph structures if connections are only made between distinct domains from $V_1, \ldots, V_2$ (cf. Sec. 3). The resulting graph is visualized using methods for multi-partite graphs, comparable to flow diagrams, like shown in [vdVvW00]: To interact in this information network we use *direct manipulation*, more precisely *drag-and-drop*, to support vague user requests. The laws of form like the *principle of proximity* can be applied to create context proximity and to map users intentions. Imaginable are user queries like in scheduling applications: *"I tend to not using this room for our meeting"* or *"It is not necessary for this special person to take part at our meeting"*. This creates *implicit context proximity* comparable to [TS00].

Furthermore visualization and interaction are integrated in the same user interface, which supports *directness*. It is left to be evaluated whether this technique enhances usability and user experience. A good evaluation result will show the benefit of direct interaction of this information network.

5 Application Example

In this chapter we discuss an application, which benefits from NoDeS as the underlying graph structure: Agent-driven event scheduling (cf. [Pay93], [PG02]).

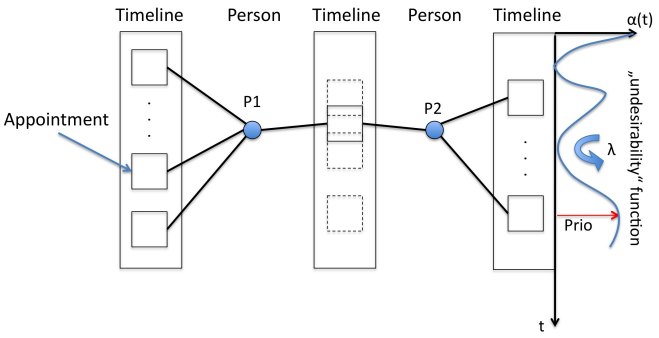


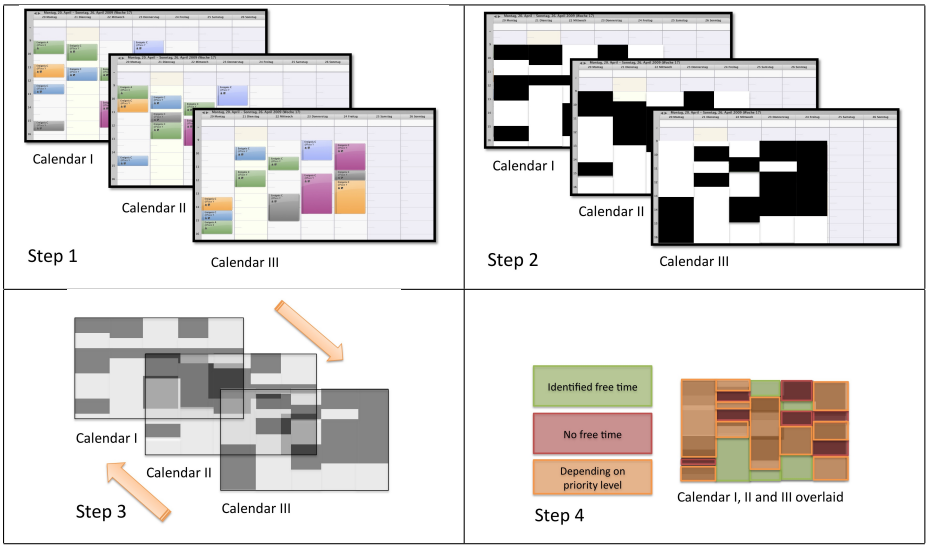
Figure 4: NoDeS used as schedule structure. Appointments are positioned along the continuous timelines but part of the graph structure and linked with the participants. The "undesirability function" steers priorities and the "leeway" between appointments.

From an application point of view, we refer to examinations of groupware aspects at the example of calendar applications like done in [GS86], on the work of Beard (cf. [BPH+90]), who proposed a visual calendar for group meetings. Transferring the single user scheduling (cf. Personal Information Management (PIM) Systems, e.g. [JT06]) into multiple user applications, we also consider Computer Supported Cooperative Work (CSCW) topics like Community Building and user management. This transfer of a single into a multiuser mode is also discussed in [Eri06]. To our knowledge there is no approach using a graph of appointments and participants to represent the scheduling problem. We show a solution which uses NoDeS to formulate event scheduling as a minimization problem.

According to [LD89] we need to consider the prospect that calendars are a special form of continuous data integration. Contrary to time measurements we do not need to care about every second. When dealing with calendar scheduling problems we deal with appointments, which have a start time, an end time and a consequential duration time. The time line can be discretized by dividing the continuous time line into subsequent slots. The length of these slots decides the solution of the calendar, i.e. the granularity in which appointments can be made. However, the discretization yields some major disadvantages:

- The granularity of the calendar has to be chosen prior to its initial creation, i.e. whenever the resolution needs to be changed, the graph structure must be recalculated.
- The graph structure is bloated by nodes representing empty slots or subsequent nodes belonging to the same appointment.

Table 1: Steps for calendar preparation: From multiple calendars to a map showing free slots.



- From a semantic point of view, the calendar graph represents a sequence of time slots, where some of these slots may have special states denoting them as real appointments, rather than nodes representing appointments themselves. Especially when linking to a distinct appointment, it is difficult to decide which of the time slots contained in this appointment should be considered the *major vertex* for this appointment, i.e. to which of the slots the edge should be connected. Solving this issue by connecting any time slot to the appointment only shifts the problem, as all slots belonging to the appointment must be evaluated to obtain its neighbors.

- When the appointment is moved, the whole link structure must be adapted, i.e. edges removed and re-inserted in other vertices in the graph. Moving an appointment should be possible by just changing its start and end time.

To remedy these disadvantages we propose a graph representation which contains only existing appointments as vertices, which can have an arbitrary start and end on the continuous time line (cf. Fig. 4). Continuous node features are tagged to the classes "Appointments", which are linked to discrete vertices like persons (e.g. P1, P2) taking part in the specific appointment. An "undesirability function" denotes the fitness of a specific time for scheduling a meeting: α represents the grade of "undesirability" of a date, where a value of zero represents a free time slot and greater values the grade to which the slot is blocked, i.e. its priority over other appointments. The slope of the graph, λ, specifies the "leeway" between appointments. The higher the value, the more will an appointment block time beyond its boundaries, leading to more free time between the specific dates. A λ-value of zero leads to subsequent appointments with no free time between them. This is

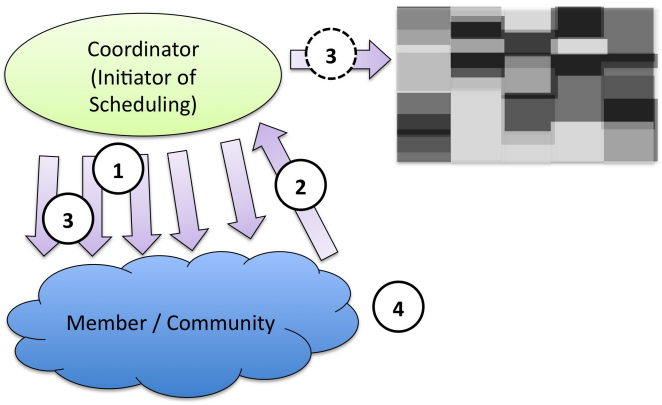


Figure 5: System architecture for scheduling application.

important to adapt vague scheduling. For optimal scheduling of a meeting the sum of the undesirability functions for all participants needs to be minimal.

Before using various calendars they first need to be prepared: By overlaying them (cf. Table 1) different steps of gray types appear, which allow a categorization of time slots: black (blocked, inevitable), gray (normal), white (free) and arbitrary fine-grained levels between them.

A possible technical realization of this special example is shown in in Fig. 5. First we assume the user maintains an online accessible calendar with priorities for each appointment. Similar to [JFR+01] and [ESTT97] our system architecture provides a central coordinator, who initiates the scheduling. Given these information the following steps will be accomplished by the system:

1. A distinct ID is generated by the system.
2. Community members send and update their calendar data.
3. Estimated time slots are published.
4. The coordinator sends the first free time slots as a request to the user community.
5. The appointment has been accepted, otherwise go back to step 2.

Further examples of possible applications can be found in physics for instance: In thermodynamics and quantum physics usual discrete iterations are thus extended into a continuous flow (cf. [AF98]).

In future work one can also consider social aspects of scheduling and group meetings like thought about in [Pal99]. Also enhancements of our scheduler in a more personalized way like done in [TGW06] are possible. For readers who would like to read up more precisely on calendar use we suggest Brush's "Taking a closer look to calendar use" ([BT05]).

6 Conclusion

In this paper we proposed a conceptual design approach towards graph structures which is able to support continuous node features, called NoDeS. We presented an application, the agent-driven event scheduling, to provide a use-case for embedding continuous information in form of time-dependent nodes into the discrete structure of a network comprising human participants and resources such as meeting rooms.

This approach can be transferred to different domains. For example, to support creative exploration and discovery by switching between contexts (BISON) or to provide visualization and exploration of huge and complex data structures in energy and logistic networks (ViERforES).

7 Acknowledgement

Stefan Haun is funded by the European Commission under the 7th Framework Programme FP7-ICT-2007-C FET-Open, contract no. BISON-211898.

Marcus Nitsche is funded by the German Ministry of Eductaion and Science (BMBF) within the ViERforES project (no. 01IM08003C).

References

[AF98] R. Aldrovandi and L. P. Freitas. Continuous iteration of dynamical maps. *Journal of Mathematical Physics*, 39(10):5324–5336, 1998.

[BPH+90] David Beard, Murugappan Palaniappan, Alan Humm, David Banks, Anil Nair, and Yen-Ping Shan. A visual calendar for scheduling group meetings. In *CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 279–290, New York, NY, USA, 1990. ACM.

[BT05] A.J. Bernheim Brush and Tammara Combs Turner. A survey of personal and household scheduling. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 330–331, New York, NY, USA, 2005. ACM.

[CH07] Diane J. Cook and Jawrence B. Holder, editors. *Mining Graph Data*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2007.

[DBETT94] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs: An annotated bibliography. *Computational Geometry: Theory and Applications*, 4(5):235 – 282, 1994.

[Die06] Reinhard Diestel. *Graphentheorie*, volume 3. Springer, Heidelberg, September 2006.

[DL07] Walter Didimo and Guiseppe Liotta. *Mining Graph Data (Graph Visualization and Data Mining)*, chapter 3, pages 35 – 63. John Wiley & Sons, Inc., Hoboken, New Jersey, 2007.

[DRS04] T. Dweyer, H. Rolletschek, and F. Schreiber. Representing experimental biological data in metabolic networks. In *2nd Asia-Pacific Bioinformatics Conference (APBC'04)*, volume 29 of CRPIT, pages 13 – 20, Sydney, 2004. ACS.

[Ead84] P. Eades. A heuristic for graph drawing. *Congr. Numer.*, 42:149 – 160, 1984.

[Eri06] Thomas Erickson. From PIM to GIM: personal information management in group contexts. *Commun. ACM*, 49(1):74–75, 2006.

[ESTT97] Keith Edwards, Mike J. Spreitzer, Douglas B. Terry, and Marvin M. Theimer. Designing and Implementing Asynchronous Collaborative Applications with Bayou, 1997.

[GS86] Irene Greif and Sunil Sarin. Data sharing in group work. In *CSCW '86: Proceedings of the 1986 ACM conference on Computer-supported cooperative work*, pages 175–183, New York, NY, USA, 1986. ACM.

[HKV06] Heiko Haller, Felix Kugel, and Max Völkel. iMapping Wikis - Towards a Graphical Environment for Semantic Knowledge Management. In *SemWiki*, 2006.

[HMM00] I. Herman, G. Melancon, and M. S. Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24 – 43, 2000.

[JFR+01] Daniel Ford Joann, Daniel A. Ford, Joann Ruvolo, Stefan Edlund, Jussi Myllymaki, James Kaufman, Jared Jackson, and Martin Gerlach. Tempus Fugit: A system for making semantic connections, 2001.

[JT06] M. Juntumaa and V.K. Tuunainen. Pim applications - an explorative study on benefits and barriers. In *Proceedings of the 19th Bled Econference: Evalues*, Bled, Slovenia, 2006.

[LD89] K. K. Gordon Lan and David L. Demets. Group sequential procedures: Calendar versus information time. *Statistics in Medicine*, 8:1191–1198, 1989.

[Pal99] Leysia Palen. Social, individual and technological issues for groupware calendar systems. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–24, New York, NY, USA, 1999. ACM.

[Pay93] Stephen J. Payne. Understanding calendar use. *Hum.-Comput. Interact.*, 8(2):83–100, 1993.

[PG02] Leysia Palen and Jonathan Grudin. Discretionary Adoption of Group Support Software: Lessons from Calendar Applications. In *In B.E. Munkvold (Ed.), Organizational*, pages 159–179. Springer Verlag, 2002.

[Tam99] R. Tamassia. Advances in the theory and practice of graph drawing. *Theoretical Computer Science*, 17:235 – 254, 1999.

[TGW06] Martin Tomitsch, Thomas Grechenig, and Pia Wascher. Personal and private calendar interfaces support private patterns: diaries, relations, emotional expressions. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, pages 401–404, New York, NY, USA, 2006. ACM.

[TS00] Albrecht Schmidt Telecooperation and Albrecht Schmidt. Implicit Human Computer Interaction Through Context. Technical report, Personal Technologies, 2000.

[vdVvW00] Gerrit van der Veer and Martijn van Welie. Task based groupware design: putting theory into practice. In *DIS '00: Proceedings of the 3rd conference on Designing interactive systems*, pages 326–337, New York, NY, USA, 2000. ACM.