HAUSHALTEGENERIERUNG: MELDEREGISTER UND GRAPHENTHEORIE

Jakob Seifert, M.Sc.

Die Haushaltegenerierung war eine Unteraufgabe des Großprojektes Zensus 2022 mit dem Ziel, Haushaltsstrukturen in der Bevölkerung abzubilden. Da Haushalte im Melderegister nicht hinterlegt sind, wurden hierzu verschiedene von Fachexperten ersonnene Heuristiken angewandt, um derartige Strukturen anhand der vorhandenen Daten erkennen zu können. Für die algorithmische Umsetzung dieser Heuristiken war eine Übersetzung in die Sprache der Graphentheorie naheliegend. Im Folgenden wird dieser Vorgang beschrieben.

Jakob Seifert, M.Sc.

Bild: Landesamt (IG Fotografie)

Jakob Seifert war vom Juli 2019 bis Juni 2024 im Bayerischen Landesamt für Statistik tätig und beschäftigte sich als Referent im Sachgebiet 24 „Verfahrensentwicklung Statistik" mit der algorithmischen Umsetzung der fachlichen Anforderung für die Haushaltegenerierung des Zensus 2022. Zuvor studierte er in Erlangen Mathematik mit Schwerpunkt Wahrscheinlichkeitstheorie und war als Aktuar bei einer namhaften Versicherungsgruppe mit Sitz in Nürnberg angestellt.

Aufgabe der Haushaltegenerierung
Überblick

Die Haushaltegenerierung ist als eines der Teilprojekte ein wichtiger Baustein des Zensus 2022. Zentrale Aufgabe der Haushaltegenerierung ist die Konstruktion von Wohnhaushalten – einer statistischen Einheit, die im Zensus nicht (vollständig) direkt beobachtet wird. Die der Haushaltegenerierung zugelieferten Daten beziehen sich auf Gemeinden, Anschriften, Gebäude, Wohnungen und Personen.

Für die Haushaltegenerierung wurden die Anschriften Deutschlands – vereinfacht ausgedrückt – in zwei Teile geteilt. Die Haushalte des ersten Teils, der knapp 10 % der betrachteten Anschriften ausmacht, wurden direkt durch einen Besuch vor Ort erhoben. Für den zweiten, deutlich größeren Teil standen jedoch ausschließlich Daten aus dem Melderegister und der Gebäude- und Wohnungszählung zur Verfügung. Diese Daten wurden genutzt, um mittels verschiedener Heuristiken die Haushaltsstrukturen zu ermitteln. Die Anwendung der Heuristiken erfolgte durch schrittweise Verarbeitung der Eingangsdaten durch neun Module, wobei mit sehr harten Indikatoren in Modul 1 begonnen wird, die nach und nach abgeschwächt und in den späteren Modulen um statistische Verfahren ergänzt werden. Weitere Informationen zur Haushaltegenerierung können auch den Bayern-in-Zahlen-Beiträgen aus 08/2024 „Trennungen und Zusammenlegungen in der Haushaltegenerierung im Zensus 2022" von Dr. Meierling und aus 07/2024 „Software-Entwicklung im Zensus-Teilprojekt Haushaltegenerierung" von Dr. Bienk entnommen werden.

Herausforderungen in der Softwareentwicklung

Neben der mathematischen Konzeption der Algorithmen zur Umsetzung der fachlich motivierten Berechnungsvorschriften bestand die größte Herausforderung in der softwaretechnischen Aufgabe, die Auswirkungen fehlerhafter oder unvollständiger Eingangsdaten auf diese Fach-Heuristiken a priori abzuschätzen und einen entsprechend fehlertoleranten Code zu erstellen, um einen reibungslosen Ablauf des Programmes sicherzustellen. Dabei war es eine besondere Herausforderung, dass aus Datenschutzgründen während der Entwicklungs- und Testphase keinerlei Zugriff auf Echtdaten möglich war, um einen Überblick über die Anzahl und die Art der Datenfehler zu bekommen. Daher wurde immer versucht, vom Worst-Case-Szenario auszugehen.

Dieses Vorgehen hat sich insofern bewährt, als dass einige dieser Randfälle dann tatsächlich in den Echtdaten aufgetreten sind. Insbesondere durch die ersten, noch nicht vollständig aufbereiteten Datenlieferungen wurde die Stabilität des Programms teilweise auf die Probe gestellt.

Die Methodik von Modul 1 der Haushaltegenerierung

Im ersten Modul der Haushaltegenerierung werden Melderegisterdaten, die auf familiäre Beziehung zwischen Personen hindeuten, untersucht. Man nimmt dann vorerst an, dass verwandte Personen in einem Haushalt leben. Werden in späteren Modulen Daten gefunden, die gegen diese These sprechen, werden entsprechende Haushalte eventuell wieder aufgetrennt. Auch hier sei wieder auf den Bayern-in-Zahlen-Beitrag aus 08/2024 von Dr. Meierling verwiesen.

Im Melderegister hat jede Person Einträge zum Ehepartner beziehungsweise eingetragenen Lebenspartner [5], zu gesetzlichen Vormündern (meistens die Eltern) und zu Kindern (siehe Infobox „Melderegister“). Steht in einem dieser Einträge nun eine andere Person mit der gleichen Anschrift, so nennt man die beiden Personen verzeigert. Verzeigerte Personen gehören stets demselben Haushalt an. Dies soll auch für mehrere Personen gelten, von denen jeweils eine mit einer anderen über mehrere direkte Verzeigerungen verbunden ist. Hat zum Beispiel eine Großmutter einen Kind-Eintrag zu ihrer Tochter, die wiederum einen Eintrag zu ihrer eigenen Tochter hat, so sollen alle drei demselben Haushalt angehören – auch wenn keine direkte Beziehung zwischen der Enkeltochter und der Großmutter ermittelt werden kann.

Je nachdem, wie genau die Übereinstimmung zwischen dem Eintrag und der anderen gefunden Person ist, soll die Verknüpfung zwischen den beiden Personen stärker oder weniger stark sein (z. B. Übereinstimmung in Ordnungsmerkmal, Name und Geburtsdatum wiegt mehr als eine Übereinstimmung nur in Nachname und Geburtsdatum).

Diese Struktur, bei der Personen paarweise miteinander durch Beziehungen verbunden sind, lädt stark zu einer mathematischen Modellierung durch einen Graphen ein.

Graphentheorie

Definitionen

Ein *(einfacher ungerichteter endlicher) Graph* im graphentheoretischen Sinne ist eine endliche Menge von *Knoten* – im Allgemeinen als Kreise dargestellt – und eine Menge von Kanten, die jeweils zwischen zwei Knoten verlaufen. Ein klassisches Beispiel wäre das Eisenbahnnetz von Deutschland, bei dem die Bahnhöfe die Knoten sind und die Schienen die Kanten. Parallele Schienen zwischen zwei Bahnhöfen werden zu einer Kante zusammengefasst, sonst handelte es sich um einen sogenannten *Multigraphen*.

Eigenschaften solcher Graphen werden seit dem 18. Jahrhundert untersucht (siehe auch Infobox „Königsberger Brücken").

Zwei Knoten heißen *benachbart*, wenn es eine Kante zwischen ihnen gibt. In Abbildung 1 wären beispielsweise die Knoten „Nürnberg Hbf." und „Fürth Hbf." benachbart.

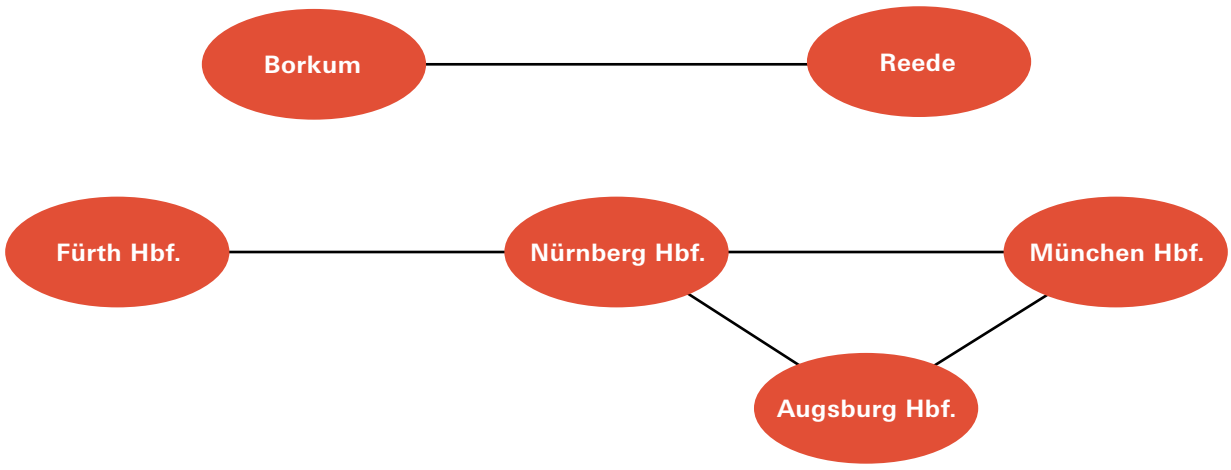
Ein *Weg* ist eine Folge von Knoten, bei der jeder Knoten höchstens einmal vorkommt und jeder Knoten der Folge mit dem nächsten Knoten der Folge durch eine Kante verbunden ist. In Abbildung 1 wäre beispielsweise „Fürth Hbf., Nürnberg Hbf., München Hbf." ein Weg.

Ein Graph heißt *zusammenhängend*, wenn es für zwei beliebige Knoten aus dem Graphen immer einen Weg gibt, dessen Anfang der eine Knoten und dessen Ende der andere Knoten ist. Der Graph in Abbildung 1 ist nicht zusammenhängend, da beispielsweise zwischen „Borkum" und „Fürth Hbf." kein Weg existiert.

Ist ein Graph nicht zusammenhängend, so zerfällt er auf eindeutige Weise in größtmögliche zusammenhängende Teilgraphen. Diese nennt man *Zusammenhangskomponenten* des Graphen. In Abbildung 1 gibt es die zwei Zusammenhangskomponenten „Borkum, Reede" und „Fürth Hbf., Nürnberg Hbf., München Hbf., Augsburg Hbf.".

Offenbar ist jeder Knoten eines Graphen Teil von genau einer Zusammenhangskomponente des Graphen.

Abbildung 1
Beispielhafter Ausschnitt des deutschen Eisenbahnnetzes als Graph



Bestimmen von Zusammenhangskomponenten

Eine typische Frage in diesem Kontext wäre nun folgende: Wie kann man algorithmisch die Zusammenhangskomponenten eines Graphen bestimmen, der eventuell sehr „komplex“ und nicht wie Abbildung 1 einfach zu überblicken ist?

Eine mögliche Antwort wäre die sogenannte Breitensuche. Dabei werden von einem beliebigen Startknoten alle Nachbarn bestimmt. Der Startknoten und alle Nachbarn werden gespeichert. Nun werden von allen Nachbarn die Nachbarn bestimmt und alle auf diese Weise neu gefundenen Knoten ebenfalls gespeichert. Von diesen neu gefundenen Knoten werden nun alle Nachbarn bestimmt und so weiter, bis irgendwann keine neuen Knoten mehr gefunden werden. Alle auf diese Weise gefundenen Knoten sind eine Zusammenhangskomponente.

Mit einem beliebigen bisher nicht gefundenen Knoten als neuen Startknoten kann man nun weiter machen, bis alle Zusammenhangskomponenten gefunden wurden.

Zuerst wollen wir nun zeigen, dass dieser Algorithmus korrekt funktioniert.

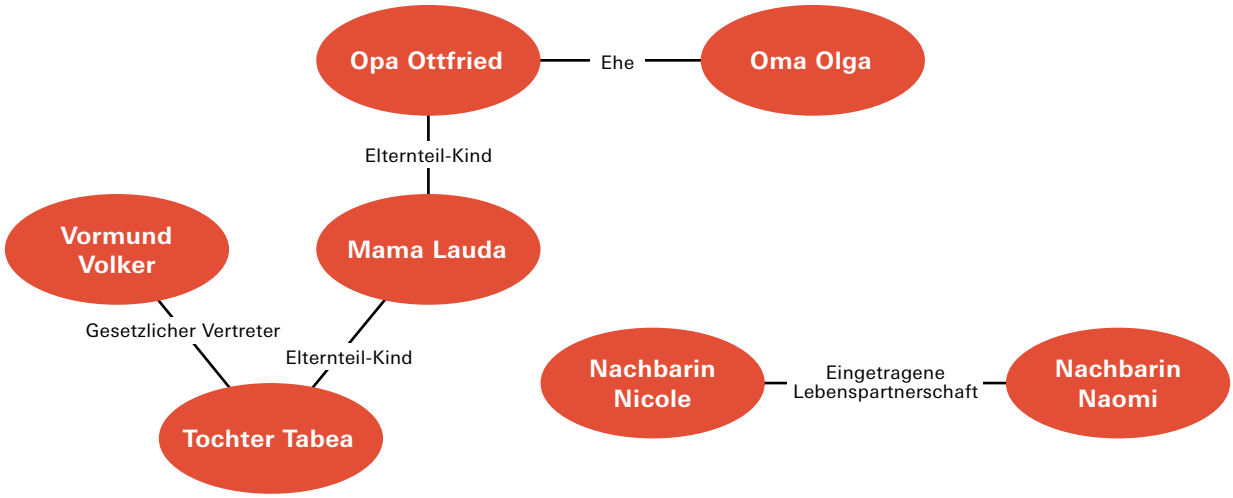
Für alle Knoten, die auf diese Weise gefunden wurden, muss es einen Weg zum Startknoten geben, da ein Weg ja gerade eine Kette von Nachbarschaften ist. Zumindest über den Umweg des Startknotens gibt es damit auch für alle Knoten untereinander einen Weg. Daher sind alle gefundenen Knoten Teil einer Zusammenhangskomponente. Umgekehrt gibt es von Knoten, die auf diese Weise nicht gefunden wurden, keinen Weg zum Startknoten und damit können sie kein Teil der Zusammenhangskomponente sein. Folglich sind die gefundenen Knoten gerade die Zusammenhangskomponente, die den Startknoten enthält.

Da es nur endlich viele Knoten gibt, kann es auch nur endlich viele Zusammenhangskomponenten geben. Weil wir mit jedem neuen Startknoten eine weitere Zusammenhangskomponente finden, müssen irgendwann alle gefunden worden sein. Damit ist gezeigt, dass der Algorithmus korrekt funktioniert.

Nachdem wir die Korrektheit bewiesen haben, sind zwei naheliegende Fragen, wie lange die Ausführung des Algorithmus dauert und wie viel Speicher man für ihn bereitstellen muss. Zu Zweiterem stellen wir fest, dass wir jeden Knoten genau einmal speichern – nämlich wenn seine Zusammenhangskomponente ermittelt wird. Damit skaliert der Speicherbedarf linear mit der Anzahl der Knoten im Graphen, das heißt, hat ein Graph doppelt so viele Knoten wie ein anderer, braucht der Algorithmus auch in etwa den doppelten Speicherplatz.

Bezüglich der Laufzeit sehen wir, dass wir jede Kante genau zweimal ablaufen: einmal beim Bestimmen der Nachbarn des einen Knoten der Kante, einmal beim Bestimmen der Nachbarn des anderen. Weiterhin müssen wir jeden Knoten einmal speichern, was ebenfalls Zeit braucht. Damit ist die Laufzeit linear zur Anzahl der Knoten plus Anzahl der Kanten. Da die Anzahl der Kanten begrenzt ist durch die Anzahl der Knoten im Quadrat (es kann maximal von jedem Knoten zu jedem Knoten eine Kante geben), könnte man alternativ auch sagen, dass die größtmögliche Laufzeit quadratisch mit der Anzahl der Knoten wächst. Das heißt, hat ein Graph doppelt so viele Knoten wie ein anderer und wir haben keine Information über die Kantenanzahlen, können wir schätzen, dass der Algorithmus im Worst Case in etwa viermal so lange läuft.

Abbildung 2
Beispielhafte Anschrift in Modul 1



Graphentheorie bei der Implementierung von Modul 1

Modellierung

Aus mathematischer Sicht kann man die Aufgabenstellung von Modul 1 folgendermaßen modellieren: Ein Graph, dessen Knoten die Personen einer Anschrift sind, und dessen Kanten die Beziehungen zwischen jeweils zwei Personen darstellen, soll in seine Zusammenhangskomponenten zerlegt werden, die dann jeweils einen Haushalt bilden sollen.

Behebung von Konflikten

Allerdings sind zuvor eventuell noch „Korrekturarbeiten“ vonnöten, da sonst wegen mangelhafter Eingangsdaten nicht vorgesehene (vgl. [8]) Beziehungskonstellationen zwischen den Personen entstehen könnten. Dazu zählen

- Personen mit mehreren Partnern,
- Personen mit mehr als zwei Eltern,
- Personen mit mehr als vier gesetzlichen Vertretern,
- Partnerschaften zwischen Großeltern und Enkeln (direkte Verwandtschaften mit größerem Generationenabstand sollten nicht untersucht werden),
- Partnerschaften zwischen Geschwistern.

Diese könnten insbesondere im weiteren Verlauf der Fachanwendung die Funktionalität der Algorithmen beeinträchtigen, wenn sie unerwartet auftreten.

> *Auch wenn intuitiv vermutet werden kann, dass solche Konflikte nur sehr vereinzelt auftreten, ist es in der Algorithmik wichtig, sich auch mit solchen Randfällen ausgiebig zu beschäftigen, um sicherzugehen, dass diese nicht zu langen Laufzeiten oder gar Endlosschleifen und Programmabbrüchen führen.*

Um solche Konfliktsituationen mit sich widersprechenden Kanten aufzulösen, wurde jeder Kante ein Gewicht gegeben. Dabei sollen Kanten, die aufgrund vieler Übereinstimmungen zwischen einer verzeigerten Person und einer Person im Melderegister gebildet wurden und deshalb plausibler erscheinen, ein höheres Gewicht haben als solche, die nur aufgrund vereinzelter Übereinstimmungen gefunden wurden. Dann wurde als Zielvorgabe angegeben, dass möglichst erst Kanten niedrigen Gewichts gelöscht werden sollen, bis die Konflikte bereinigt sind. Also wurde beispielsweise, falls ein Kind drei Kanten zu möglichen Elternteilen hatte, die Kante zu demjenigen Elternteil gelöscht, die das geringste Gewicht hatte.

Dem aufmerksamen Lesenden wird nicht entgangen sein, dass zum Erkennen solcher Konflikte auch die Richtung der Elternteil-Kind- beziehungsweise Vormund-Mündel-Kanten wichtig ist, das heißt, die Information, welche Person die Rolle des Elternteils beziehungsweise des Vormunds einnimmt. Daher wurde für die Modellierung in Modul 1 tatsächlich ein gerichteter Graph verwendet. Aus Gründen der Übersichtlichkeit sind wir aber bisher nicht weiter auf dieses Detail eingegangen.

Auch wenn intuitiv vermutet werden kann, dass solche Konflikte nur sehr vereinzelt auftreten, ist es in der Algorithmik wichtig, sich auch mit solchen Randfällen ausgiebig zu beschäftigen, um sicherzugehen, dass diese nicht zu langen Laufzeiten oder gar Endlosschleifen und Programmabbrüchen führen. Im finalen Eingabedatenbestand gab es noch etwa 1 500 größere Konfliktcluster (mehr als zehn zusammenhängende Kanten, die sich gegenseitig widersprechen), in den vorläufigen Lieferungen jedoch wesentlich mehr.

Behebung von widersprüchlichen Partnerschaftskanten via Matching

Wir wollen nun den ersten möglichen Konflikt näher betrachten: Eine oder mehrere Personen haben mehrere Partnerschaftskanten. Wie oben beschrieben wäre nun der erste Schritt, nach und nach die Kanten niedriger Übereinstimmung und somit niedrigen Gewichts zu löschen, bis der Konflikt behoben ist. Was aber, wenn wir in eine Situation geraten, in der alle Kanten das gleiche Gewicht haben? Es sei noch einmal betont, dass – solange eine derartige Konstellation theoretisch irgendwie möglich ist – es dringend geboten ist, sich mit den Konsequenzen daraus zu beschäftigen.

Welche Lösungsoptionen gibt es also?

Eine Möglichkeit wäre dann, einfach alle Kanten zu löschen; allerdings würde man damit auch jede Information über mögliche Beziehungen zwischen den Personen verlieren. Jede Kante hat ja durchaus eine Existenzberechtigung in dem Sinne, dass es Indizien gibt, die für eine Partnerschaft zwischen ihren beiden Personen sprechen.

Daher ist auch ein zweiter denkbarer Ansatz, nämlich zufällig solange Kanten zu löschen, bis alle Konflikte behoben sind, suboptimal. Auch dabei kann es leicht passieren, dass mehr Information als nötig verworfen wird.

Ziel muss es also sein, alle Konflikte zu beheben und dabei möglichst viel Information, also möglichst viele Kanten, zu behalten. Dies ist mathematisch äquivalent zum sogenannten Matching-Problem.

Beim Matching geht es darum, möglichst viele Paare von benachbarten Knoten eines gegebenen Graphen zu bilden, wobei jeder Knoten nur in einem Paar auftreten darf. Ein Anwendungsfall wäre das Zuteilen von Aufgaben unter Mitarbeitenden, die unterschiedliche Fähigkeiten haben. Mit solchen Problemen beschäftigt sich die Mathematik-Gemeinde seit Ende des 19. Jahrhunderts [6].

Der erste Lösungsalgorithmus für allgemeine Matching-Probleme, dessen Laufzeit polynomiell mit der Anzahl der Knoten wächst, wurde 1965 von Jack Edmonds gefunden [1]. Dieser Algorithmus hat eine Laufzeit, die mit der dritten Potenz der Knotenanzahl skaliert [4]; das heißt, hat ein Graph doppelt so viele Knoten wie ein anderer, braucht der Algorithmus zum Finden eines optimalen Matchings in etwa achtmal so lange wie für den anderen.

Mit diesem Algorithmus lässt sich nun das Problem widersprüchlicher Partnerschaftskanten lösen. Allerdings kann es sein (und ist in den Echtdaten wie oben geschrieben auch vorgekommen), dass zusätzlich einer der anderen oben genannten Konflikte auftritt, und deshalb eine Lösung nur für das Matching-Problem nicht zielführend für die Auflösung aller Konflikte ist. Für solche Fälle wurden in der Haushaltegenerierung ausgefeiltere Spezialalgorithmen entwickelt, von denen ein letzter noch vorgestellt werden soll.

Behebung von verbotenen Partnerschaften

Wir wollen nun die beiden letzten Konfliktarten betrachten, mögliche mit dem Gesetz in Konflikt stehende inzestuöse Beziehungen [8]. Dankenswerterweise kamen solche in der letzten Echtdatenlieferung nicht mehr vor, und die Überlegungen hierzu waren daher a posteriori überflüssig. Sie bieten aber Anlass, ein beispielhaftes graphentheoretisches Konstrukt zu demonstrieren.

Da im Melderegister weder Großelternteil-Enkelkind-Beziehungen noch Geschwisterbeziehungen hinterlegt sind, erkennen wir diese Beziehungen zwischen zwei Personen nur indirekt über eine Kette von zwei Elternteil-Kind-Kanten, die in der Fachanwendung Nachkommen-Kanten (NK-Kanten) genannt werden, oder zwei Elternteil-Kind-Kanten von einem Elternteil aus. Man sieht also, dass solche Partnerschaften immer Teil eines Dreiecks zwischen zwei NK-Kanten und einer Partnerkante sind.

Wir betrachten hierzu nun auch gerichtete Kanten, denen durch einen Pfeil eine Orientierung gegeben wird (siehe Abbildung 3). Diese Dreiecke lassen sich immer durch das Löschen einer Kante in eine valide Beziehungsform auflösen. Da dabei weiterhin alle Personen in einer Zusammenhangskomponente verbleiben, ändert das Löschen einer Kante auch nichts an den gebildeten Haushalten.

Wie weiter oben geschrieben, wird dabei immer zuerst die Kante mit dem niedrigsten Gewicht gelöscht. Spannend ist also nur der Fall, dass alle NK-Kanten das gleiche Gewicht haben. (Aufgrund der Modellierung können Partnerschaftskanten nicht das gleiche Gewicht wie NK-Kanten haben. Daher genügt es, sich die NK-Kanten anzusehen, da entweder die Partnerkante eines Dreiecks ein niedrigeres Gewicht hat, und dann vor den NK-Kanten gelöscht wird, oder ein höheres, und dann eine der beiden NK-Kanten vor der Partnerkante gelöscht wird.)

Abbildung 3
„Verbotene“ Dreiecke

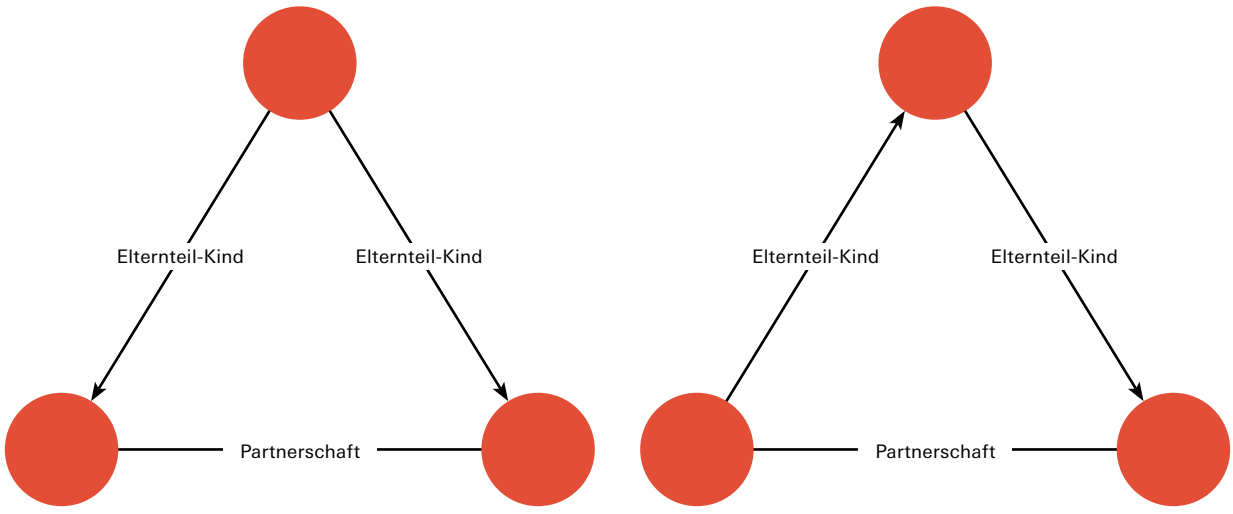
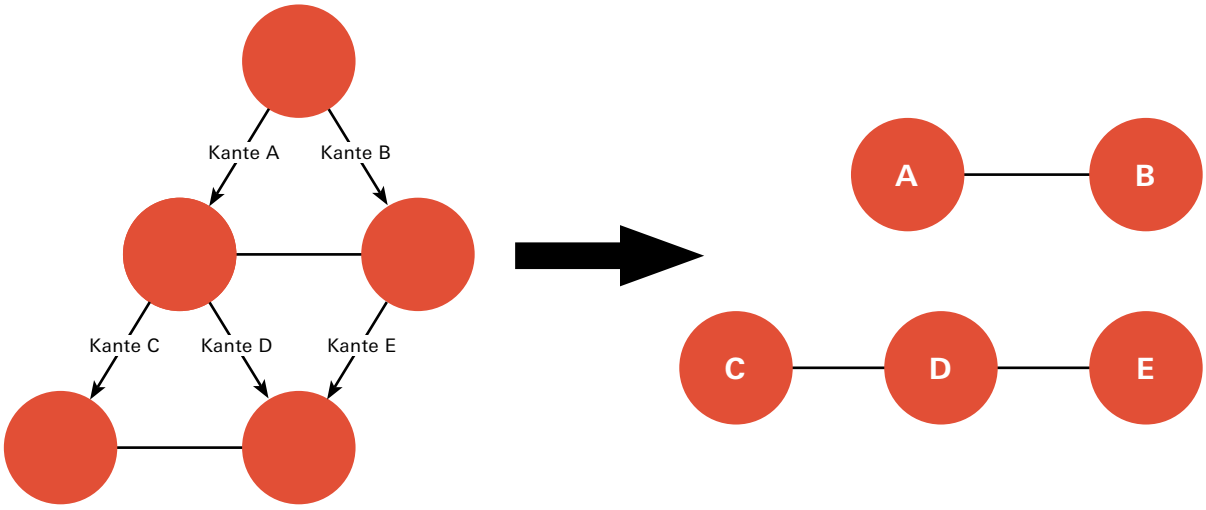


Abbildung 4
Graph und Meta-Graph



Sind die NK-Kanten nun isoliert von anderen Kanten, ist es aus Sicht der maximalen Informationsbewahrung egal, welche der beiden gelöscht werden. Theoretisch aber könnten eine oder beide der NK-Kanten noch Teil anderer verbotener Partnerschaftsbeziehungen sein. Dann wäre wieder die Frage, wie man alle diese Konflikte auflöst, während man gleichzeitig möglichst wenige Kanten löscht.

Da die NK-Kanten ja immer Paare in einem Dreieck bilden, kann man das Problem nun mit folgendem „Trick" modellieren:

Die NK-Kanten sind die Knoten eines Meta-Graphen, von denen zwei mit einer Kante verbunden sind, genau dann, wenn sie Teil eines solchen Dreieckes sind (siehe Abbildung 4). Ziel ist es dann, möglichst wenige Knoten aus dem Meta-Graphen zu löschen, sodass von allen Kanten mindestens ein Endknoten gelöscht wurde. Dieses Problem heißt *Knotenüberdeckung*.

Im Allgemeinen gibt es für diese Problemstellung keine effiziente algorithmische Lösung, das Problem ist NP-vollständig [2]. Das bedeutet (falls die Vermutung richtig ist [7]), im schlimmsten Fall wächst die Laufzeit eines Lösungsalgorithmus schneller als jedes Polynom, wenn die Problemgröße steigt. Da in unserem Fall der Graph wegen der Art seiner Konstruktion als Modellierung von Beziehungsdreiecken jedoch eine sehr spezielle Struktur hat, findet der entwickelte Algorithmus für fast alle Fälle (mit Ausnahme extremer Randfälle dieser ohnehin bereits sehr ungewöhnlichen Konstellationen) optimale Lösungen in guter Laufzeit.

Fazit

Die Haushaltegenerierung des Zensus 2022 ist ein gelungenes Beispiel, wie in Forschungsbereichen und Anwendungen auftretende Probleme mithilfe der Mathematik, in diesem Fall der Graphentheorie, angegangen werden können.

Melderegister

Die Personendaten des Melderegisters wurden der Haushaltegenerierung mittels einer PERSON genannten Datenbanktabelle übergeben. In dieser gibt es für jeden Personeneintrag eine Zeile. Dort stehen unter anderem Ordnungsnummer, Name und Geburtsdatum der Person selber, als auch für deren bis zu vier Vertreter, bis zu 20 Kinder und möglichen Partner. Es wurde sich gegen eine Auftrennung in mehrere Tabellen entschieden, sodass die PERSON-Tabelle recht groß und unübersichtlich war.
Da der Zugriff auf Echtdaten für Entwickler aus Datenschutzgründen streng reglementiert war, wurden zu Testzwecken ein Personengenerator entwickelt, der zufällige künstliche Personeneinträge mit Fehlern erzeugt.
In den Abbildungen 5 – 7 wurde die PERSON-Tabelle aus Gründen der Übersichtlichkeit verkürzt und in drei Tabellen aufgeteilt dargestellt. Hier sieht man beispielsweise eine Verzeigerung von Maximilian Krause zu seiner Mutter Silvia Krause, die wiederum eine Verzeigerung zu Maximilian als ihren Sohn hat. Der Eintrag <null> entspricht einem fehlenden Wert.

Abbildung 5
Auszug aus der PERSON-Tabelle mit zufällig erzeugten Personendaten und Partnerdaten

	OM...	VORNAME	FAMNAME	GEBDATUM_MASTER	FAMSTAND_MASTER	OMP_PARTNER_MR	VORNAME_PARTNER_MR	FAMNAME_PARTNER_MR	GEBDATUM_PARTNER_MR
1	<null>	Matthias	Neumann	20220106	LD	<null>	<null>	<null>	<null>
2	<null>	Gabriele	Herrmann	19631205	VH	P0000031321	<null>	Herrmann	19621008
3	P0000006233	Christina	Wolf	20000715	LD	<null>	<null>	<null>	<null>
4	P0000012324	<null>	Müller	19770702	VH	<null>	Manfred	<null>	19770116
5	P0000030133	Klaus	Müller	20011107	<null>	<null>	<null>	<null>	<null>
6	P0000031321	<null>	Herrmann	19621008	VH	P0000035125	<null>	Hoffmann	19590820
7	P0000033209	<null>	Müller	20220220	LD	<null>	<null>	<null>	<null>
8	P0000035125	<null>	Hoffmann	19590820	LD	P0000031321	<null>	Herrmann	19621008
9	P0000041270	Pippilotta Viktualia Rullgardina Krusmynta Efraimsdotter	Långstrump	20120222	LD	<null>	<null>	<null>	<null>
10	P0000045336	Bernd	<null>	19760823	LD	<null>	<null>	<null>	<null>
11	P0000063878	Barbara	Neumann	20220511	LD	<null>	<null>	<null>	<null>
12	P0000068281	Mia	Herrmann	19950302	LD	<null>	<null>	<null>	<null>
13	P0000068444	Efraim fordom havets skräck numera söderhavskung	Långstrump	19840116	EA	<null>	<null>	<null>	<null>
14	P0000068612	Silvia	Krause	19971007	LD	<null>	<null>	<null>	<null>
15	P0000071658	Maximilian	Krause	20220310	LD	<null>	<null>	<null>	<null>
16	P0000077107	Manfred	<null>	19770116	VH	P0000012324	<null>	Müller	19770702
17	P0000089144	Michael	Krause	20151102	LD	<null>	<null>	<null>	<null>
18	P0000101753	Ralf	Köhler	19861226	GS	<null>	<null>	<null>	<null>

Abbildung 6
Auszug aus der PERSON-Tabelle mit zufällig erzeugten Personendaten und Vertreterdaten

	OMP_MR	VORNAME	FAMNAME	GEBDATUM_MASTER	OMP_VERTRETER1_MR	VORNAME_VERTRETER1_MR	FAMNAME_VERTRETER1_MR	GEBD
1	<null>	Matthias	Neumann	20220106	P0000011875	Sofie	Neumann	199106
2	<null>	Gabriele	Herrmann	19631205	<null>	<null>	<null>	<null>
3	P0000006233	Christina	Wolf	20000715	P0000045336	Bernd	<null>	197608
4	P0000012324	<null>	Müller	19770702	<null>	<null>	<null>	<null>
5	P0000030133	Klaus	Müller	20011107	P0000012324	<null>	Müller	197707
6	P0000031321	<null>	Herrmann	19621008	<null>	<null>	<null>	<null>
7	P0000033209	<null>	Müller	20220220	P0000030133	Klaus	Müller	200111
8	P0000035125	<null>	Hoffmann	19590820	<null>	<null>	<null>	<null>
9	P0000041270	Pippilotta Viktualia Rullgardina Krusmynta Efraimsdotter	Långstrump	20120222	P0000068444	Efraim fordom havets skräck numera söderhavskung	Långstrump	198401
10	P0000045336	Bernd	<null>	19760823	<null>	<null>	<null>	<null>
11	P0000063878	Barbara	Neumann	20220511	P0000011875	Sofie	Neumann	199106
12	P0000068281	Mia	Herrmann	19950302	<null>	Gabriele	Herrmann	196312
13	P0000068444	Efraim fordom havets skräck numera söderhavskung	Långstrump	19840116	<null>	<null>	<null>	<null>
14	P0000068612	Silvia	Krause	19971007	<null>	<null>	<null>	<null>
15	P0000071658	Maximilian	Krause	20220310	P0000068612	Silvia	Krause	199710
16	P0000077107	Manfred	<null>	19770116	<null>	<null>	<null>	<null>
17	P0000089144	Michael	Krause	20151102	<null>	<null>	<null>	<null>
18	P0000101753	Ralf	Köhler	19861226	P0000041270	Pippilotta Viktualia Rullgardina Krusmynta Efraimsdotter	Köhler	195504

Abbildung 7
Auszug aus der PERSON-Tabelle mit zufällig erzeugten Personendaten und Kinddaten

	OMP_MR	VORNAME	FAMNAME	GEBDATUM_MASTER	OMP_KIND1_MR	VORNAME_KIND1_MR	FAMNAME_KIND1_MR	GEBDATUM_KIND
1	<null>	Matthias	Neumann	20220106	<null>	<null>	<null>	<null>
2	<null>	Gabriele	Herrmann	19631205	P0000068281	Mia	Herrmann	19950302
3	P0000006233	Christina	Wolf	20000715	<null>	<null>	<null>	<null>
4	P0000012324	<null>	Müller	19770702	P0000030133	Klaus	Müller	20011107
5	P0000030133	Klaus	Müller	20011107	P0000033209	<null>	Müller	20220220
6	P0000031321	<null>	Herrmann	19621008	<null>	<null>	<null>	<null>
7	P0000033209	<null>	Müller	20220220	<null>	<null>	<null>	<null>
8	P0000035125	<null>	Hoffmann	19590820	<null>	<null>	<null>	<null>
9	P0000041270	Pippilotta Viktualia Rullgardina Krusmynta Efraimsdotter	Långstrump	20120222	<null>	Ralf	<null>	19861226
10	P0000045336	Bernd	<null>	19760823	P0000006233	Christina	Wolf	20000715
11	P0000063878	Barbara	Neumann	20220511	<null>	<null>	<null>	<null>
12	P0000068281	Mia	Herrmann	19950302	<null>	<null>	<null>	<null>
13	P0000068444	Efraim fordom havets skräck numera söderhavskung	Långstrump	19840116	P0000041270	Pippilotta Viktualia Rullgardina Krusmynta Efraimsdotter	Långstrump	20120222
14	P0000068612	Silvia	Krause	19971007	P0000071658	Maximilian	Krause	20220310
15	P0000071658	Maximilian	Krause	20220310	<null>	<null>	<null>	<null>
16	P0000077107	Manfred	<null>	19770116	<null>	<null>	<null>	<null>
17	P0000089144	Michael	Krause	20151102	<null>	<null>	<null>	<null>
18	P0000101753	Ralf	Köhler	19861226	<null>	<null>	<null>	<null>

Königsberger Brücken

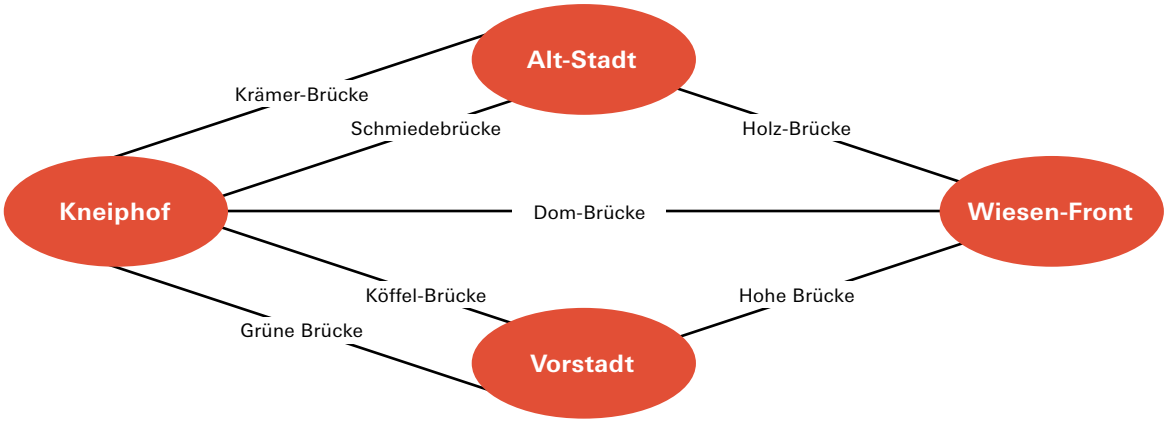
Als Geburtsstunde der Graphentheorie gilt das sogenannte Königsberger Brückenproblem. Dabei geht es um die Frage, ob es im Königsberg (dem heutigen Kaliningrad) des 18. Jahrhunderts einen Weg gab, bei dem die sieben Brücken der Stadt alle genau einmal überquert wurden (ohne zwischendurch die Flüsse auf andere Art zu überqueren). Leonhard Euler bewies dabei 1736, dass dies nicht möglich ist [3]. Dabei erkannte er, dass die genauen Abstände der Brücken untereinander genau wie die exakten Wege, die auf den vier Landstücken zurückgelegt wurden, irrelevant waren und man die Landstücke deshalb einfach als Punkte modellieren kann. Diese sind damit die Knoten eines Graphen, die durch die als Kanten modellierten Brücken verbunden sind. Da es in diesem Fall mehr als eine Kante zwischen zwei Knoten geben kann, handelt es sich um einen Multigraphen.

Die Frage ist dann, ob es möglich ist, alle Kanten des Graphen nacheinander abzulaufen, ohne eine Kante doppelt zu durchlaufen. Allgemein kann man zeigen, dass dies bei einem beliebigen Graphen genau dann möglich ist, wenn es höchstens zwei Knoten mit einer ungeraden Anzahl an Kanten gibt (das wären dann Start- und Endknoten des potenziellen Pfades) [3]. Insbesondere beim Königsberger Brückenproblem ist es also unmöglich.

Abbildung 8
Stadtplan von Königsberg von 1905

Bild: https://commons.wikimedia.org/wiki/File:K%C3%B6nigsberg_Karte_1905.jpg, abgerufen am 10.04.2024

Abbildung 9
Graphentheoretische Modellierung des Brückenproblems



Literaturverzeichnis

[1] Edmonds, J. [1965]: Paths, trees, and flowers, Canadian Journal of Mathematics 17, 449–467

[2] Karp R.M. [1975]: On the Computational Complexity of Combinatorial Problems, Networks Volume 5, Issue 1, 45–68, di: 10.1002/net.1975.5.1.45

[3] Korte B., Vygen J. [2018]: Combinatorial Optimization, Algorithms and Combinatorics 21, 6. Edition, Theorem 2.24., Springer-Verlag GmbH Germany, doi:10.1007/978-3-662-56039-6_2

[4] Korte B., Vygen J. [2018]: Combinatorial Optimization, Algorithms and Combinatorics 21, 6. Edition, Theorem 10.31., Springer-Verlag GmbH Germany, doi:10.1007/978-3-662-56039-6_10

[5] Lebenspartnerschaftsgesetz vom 16. Februar 2001 (BGBl. I S. 266), zuletzt geändert durch Artikel 7 Absatz 6 des Gesetzes vom 31. Oktober 2022 (BGBl. I S. 1966)

[6] Petersen J. [1891]: Die Theorie der regulären graphs, Acta Mathematica. 15. Jahrgang, 193–220, doi:10.1007/BF02392606

[7] www.claymath.org/millennium/p-vs-np/ abgerufen am 10.05.2024

[8] www.gesetze-im-internet.de/bgb/__1307.html abgerufen am 10.05.2024