Графи и анализ на надежността при конструиране на информационни връзки

Танчо Яков

Graph theory and analysis of reliability of information when constructing links: *Is considered a method of analysis for the design of IT links between the objects involved in a komunikatsionnen process through application of methods of mathematical modeling and operations research with application of algorithms from graph theory.*

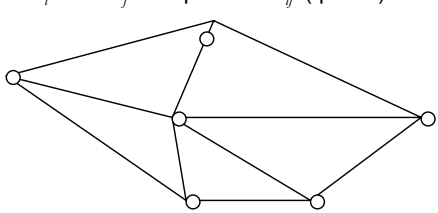
Key words: *graph, modelling, informatics, links*

ВЪВЕДЕНИЕ

При обмен на информация между обекти от дадена област, между които съществуват информационни връзки, естествено възниква въпросът как и по какъв начин да бъдат организирани връзките между отделните обекти, така че рискът от евентуална загуба, разсейване или увреждане на предаваната информация между отделните обекти да бъде сведен до минимум. Поставяме и условието заедно с това при обмена на информация да се запази изискването за достижимост на информацията между отделните обекти, т.е. между произволни два обекта да съществува връзка за обмен на информацията, или по друг начин казано да няма изолиран обект. И още едно условие - връзката между произволни два обекта да е еднозначна, т.е. между два кои да са обекта да съществува само един вариант на връзка между тях. В изложението представяме един подход с ползване на методи и алгоритми от областта на математическото моделиране и изследване на операциите с приложение на графите, който може да бъде използван в редица конкретни случаи за анализ и констуиране на връзки при комуникация между различни субекти.

1. МАТЕМАТИЧЕСКА ПОСТАНОВКА

Да си представим, че субектите участвуващи в комуникационния процес са върхове на граф, а ребрата в този граф са възможните връзки между тях. Да означим тези ребра с (x_i, x_j),което означава връзка между субектите x_i и x_j. Както споменахме по горе ние ще се стремим да намерим такъв начин за комуникиране между отделните субекти, така че вероятността или възможността от “увреждане” на информацията или нейното разсейване да е минимална. Да съпоставим на всяка връзка между субектите x_i и x_j едно число τ_{ij}, което в най-общия случай ще означавава, че вероятността да бъде “увредена” информацията при предаването й от x_i към x_j и обратно е τ_{ij} (фиг. 1).



(фиг. 1)

Така нашата задача е да се намери възможно оптималния набор от ребра в графа, като се изпълнява условието за достижимост между отделните върхове,

всички върхове в графа са включени, но при минимален риск от "увреждане" на информацията, т,е, търси се:

$$\min\{1-\prod(1-\tau_{ij})\}$$

където τ_{ij} както определихме по-горе е вероятността за увреждане или разсейване на иформацията при нейното предаване от x_i към x_j . Въведените условия показват, че съществува поне един път за достигане на информацията между всички участници, т.е. не се появява изолиран връх в графа.

Да означим с *G=(X,A*) графа на връзките, който представлява един свързан и неориентиран граф. На всяко ребро (x_i, x_j) от графа *G* да съпоставим по едно число c_{ij}, което в най-общия случай наричаме тегло на реброто (x_i, x_j) , и нека броят на тези ребра е *n*. Така всяко подмножество *G'* на графа *G=(X,A),* ще наричаме дърво на графа *G* и ще го означаваме с *T*, ако за него се изпълняват условията:

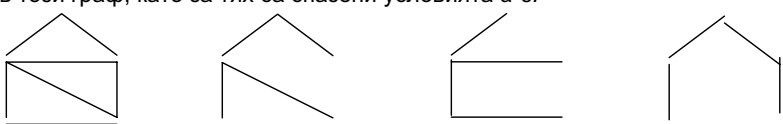
а. графа T съдържа n на брой върхове и поне n-1 ребра от множеството $\{(x_i, x_j)\}$*;*

б. няма цикли в графа T;

в. всички възможни двойки върхове са свързани помежду си само с една връзка помежду им, или по друг начин казано , съществува само един път по който може да се достигне от връх x_i *до връх* x_j *.*

Така скелет на графа *G*, ще наричаме такова дърво, което изпълнява условията *а-в*.

На фиг. 2 са дадени един граф (а) и някои от дърветата (b1), (b2). (b3) породени в този граф, като за тях са спазени условията *а-в.*



(a) (b1) (b2) (b3)

(фиг. 2)

От тук можем да въведем и понятието минималност на свързаност на графа *G=(X,A)*, т.е. граф *G'*$\subset$ *G(X,A)*, който изпълнява условията а-в, но при условието че c_{ij} е тегло на реброто (x_i, x_j) да се намери:

$$F = \min \sum_{i,j}^{n} c_{ij} \qquad (1)$$

От теорията на графите [3] понятието скелет на граф доста често се заменя с понятието обхващащо дърво, като в нашия случай ще използваме и двете. От там също е известно че ако имаме в общия случай един неориентиран граф *G* с брой на върховете *n* и графа е напълно свързан , то броят на обхващащите дървета на такъв граф е n^{n-2}. Ясно е от тук, че с нарастването на броя на дъгите в един граф , много бързо нараства броят на обхващите дървета на графа. За графа от рис.2 лесно могат да бъдат построени всички обхващащи дървета като техният брой при така зададения граф е 21. Така при нашата задача която сме си поставили, в най-общия случай да намерим от цялото множество обхващащи дървета на графа, такова което изпълнява условието (1), то е видно един такъв процес изисква достатъчно голям обем изчислителни ресурси и време, за решаването на тази задача. Извършени са изследвания и са построени някои алгоритми [2] които могат да бъдат прилагани за успешно решаване на тази задача, с оглед на наличните изчислителни ресурси, като се смята че в общия случай тази задача може да бъде

решена [5]. По долу е представен един алгоритъм, който по своята си същност е един алгоритъм за неявно претърсване на множеството от всички обхващащи дървета на граф.

2. АЛГОРИТМИ ЗА НАМИРАНЕ НА ВСИЧКИ ОБХВАЩАЩИ ДЪРВЕТА

Нека *G=(X,A)* е един неориентиран граф, и на всяко ребро (x_i, x_j) от графа да означим с едно число от *1 до m*, представляващо неговият номер, като *m* е броят на ребрата в графа. Нека също с *d** да означим степента на връх *x**, което от теорията на графите означава броят на изхождащите от върха *x** ребра (x^*, x_j).

2.1. *Обща схема на алгоритъма.* Основните стъпки на нашият алгоритъм се заключават в следното:

- избор на ребро от графа, което с до момента избраните ребра образува дърво, т.е. избраното ребро има общ връх с някое от до момента избрани ребра, което изпълнява условието за свързаност;
- за всяко избрано на предишния етап ребро се проверява дали неговият избор не води до появата на цикъл в графа, т.е. дали не се нарушава условието за единственост на достижимостта от един връх до друг, като ако избраното ребро води до появата на цикъл се избира друго ребро от множеството в порядък на нарастване на техните номера;
- ако избраното ребро не води до появата на цикъл, то се допълва множеството от избрани ребра до момента;
- проверява списъка от ребра дали не е изчерпан, като ако е изчерпан, то това означава че в конструираното ново множество от ребра са включени такива ребра, които пораждат граф с дървовидна структура и за него са изпълнени условията в т.1 *a-b*.

2.2. *Поетапно описание на алгоритъма.* Да изберем един връх *x** със степен *d**. Да разделим нашият граф *G=(X,A)* на две части по следният начин: в множеството *T* са всички ребра, имащи общ връх *x**, и множество *T*'* от ребра които нямат обш връх с *x**, т.е. получават се две редици от ребра:

- $a_1, a_2, ...a_{d^*}$ с общ връх *x**
- $a_{d^*+1}, a_{d^*+2}, ...a_m$ нямащи връзка с *x**

като *m* е броят на ребрата в графа.

<u>СТ1.</u> На всеки връх $x_i \subset X$ да съпоставим двойката (r_i, p_i), наречена двойка на предшествениците, като първият елемент от тази двойка r_i показва корен на избираното дърво, в което се съдържа върха x_i, а вторият елемент p_i ще указва върхът който се явява предшественик на върха x_i.Така първоначално $r_i = x_i$ и $p_i = 0$. Нека с *k=1* да означим брояча на ребрата.

<u>СТ2.</u> От множеството ребра избираме произволно ребро $a_k = (x_i, x_j)$, като се проверява брояча *k*. В случай че *k=m+1* , т.е. изчерпан е списъка от ребра то се преминава на СТ6, в противен случай при $k \le m$ се преминава на стъпка СТ3.

<u>СТ3.</u> Проверява се дали за избраното ребро $r_i = r_j$. В случай че $r_i = r_j$, то това означава че съответните върхове x_i и x_j са от едно и също потдърво *T* и добавяйки това ребро ще се образува цикъл в *T*. Отхвърляме това ребро a_k , полагаме *k=k+1* и се връщаме на стъпка СТ2. В случай че $r_i \ne r_j$ то добавяме избраното ребро a_k към дървото Т, чрез нарастване на списъка от избрани ребра и се преминава на СТ4.

СТ4. Свързваме поддървото *T* и новопороденото с корени съответно r_i и r_j, като изпълняваме следната процедура за промяна на двойките (r_i, p_i): заменяме $r_j = r_i$, а $p_j = x_i$.

СТ5. Избирайки *n-1* на брой ребра, където *n* е броят на върховете в графа G, ние сме получили едно обхващащо дърво в графа. Образуваното дърво *T** го запомняме и преминаваме на СТ6. Но в случай че броят на избраните до момента ребра е по малък от *n-1* то полагаме *k=k+1* и се връщаме на СТ2.

СТ6. На тази стъпка се осъществява връщане назад, като от сформираното дърво *T** премахваме последното ребро включено в него при действието на алгоритъма. Така ако нашето ребро е a_l и *l=d**, където *d** е степента на началния връх, то спира работата на алгоритъма. Така за произволно избран начален връх от графа са построени всички обхващащи дървета в графа. Но в случай че $l \neq d^*$ то е необходимо да се преобразува двойката (r_i, p_i), като се спазва следната процедура за промяната й: ако например се маха реброто (x_v, x_w) и при $p_i = x_v$ в първоначалното дърво двойката (r_i, p_i) остава непроменена, като тя се променя в последващото построявано дърво: ако обаче $p_i = x_w$ то се извършва обратната операция – в първоначалното дърво двойката (r_i, p_i) се променя, а в последващото образувано остава непроменена. След промяната на тези двойки (r_i, p_i) ние сме готови да изберем ново ребро, т.е. прменяме *k=l+1* и преминаваме на СТ2.

3. МИНИМАЛНО ОБХВАЩАЩО ДЪРВО

Нека G=(X,A) е един свързан и неориентиран граф с ребра (x_i, x_j). За тях са известни теглата и да ги означим с c_{ij}. Поставяме си следната задача: от множеството от всички обхващащи дървета да се намери такова дърво за което сумата от теглата на ребрата да е минимална, т.е. както посочихме в началото на нашата работа търсим:

$$z = \min\{1 - \prod(1 - \tau_{ij})\} \qquad (2)$$

В т.2 беше описан един възможен алгоритъм за намиране на всички обхващащи дървета в графа. Ще покажем сега че този алгоритъм се явява един добър инструмент за решаването на нашата задача, като в процеса на работа на алгоритъма при замяната на едно избрано ребро (x_i, x_j) с друго $(x_i, x_j)*$ се следи за намалянето на тяхното тегло.

СТ0. Начало на алгоритъма. За графа G=(X,A) полагаме за начало множеството на избрани върхове $T_s = \{x_s\}$, където върхът x_s е избран произволно, а множеството избрани ребра от графа полагаме на A_s, като при стартиране на алгоритъма това множество е празно, т.е. $A_s = \mathrm{O}$.

СТ1. За всички върхове на графа G, такива че $x_j \notin T_s$, т.е. те не са до момента избрани, прилагаме следната процедура. Търсим връх α_j от множеството T_s, такъв че да удовлетворява условието:

$$c(\alpha_j, x_j) = \min[c(x_i, x_j)] = \beta_j; x_s \in T_s \qquad (3),$$

при $\alpha_j \in T_s$ и за върха x_j образуваме двойката на предшествениците (α_j, β_j). В противен случай ,ако не успеем да намерим такъв връх α_j то на x_j съпоставяме двойката $(0, \infty)$.

СТ2. От множеството върхове $x_j \notin T_s$ търсим сега такъв връх $x*_j$ за който $\beta*_j = \min[\beta_j]$ при $x_j \notin T_s$ Намереният връх го включваме в множеството T_s и $T_s = T_s \bigcup \{x*_j\}$. И за съответно избраното ребро с начало $\alpha*_j$ и край $x*_j$ обновяваме множеството от избрани ребра A_s като $A_s = A_s \bigcup \{(\alpha*_j, x*_j)\}$. За избрания до момента брой прикрепени върхове в T_s проверяваме дали сме достигнали общият брой на върховете в графа G. Това показва дали всички върхове в графа са свързани в контсруираното до момента дърво, т.е. не се нарушава уловието за свързаност и достижимост между отделните върхове в графа G. Ако всички върхове са свързани то в множеството от ребра A_s се намира търсеното обхващащо дърво. Алгоритъма приключва своята работа.

СТ3. В случай че не са изчерпани всички върхове в графа G търсим между всички върхове x_j такъв че $x_j \notin T_s$ и $x_j \in G(x*_j)$, като изпълняваме следната процедура за замяна на двойката на предшествениците:

А. В случай че $\beta_j > c(x*_j, x_j)$ се полага $\beta_j = c(x*_j, x_j)$ и $\alpha_j = \alpha*_j$ и се преминава на СТ2.

Б. При $\beta_j \le c(x*_j, x_j)$ се преминава направо на СТ2.

4.ИЗЧИСЛИТЕЛНИ АСПЕКТИ, ИЗВОДИ И ПРАКТИЧЕСКИ ПРИЛОЖЕНИЯ

Както бе отбелязано в т.2 при напълно свързан граф G=(X,A) с брой на върхове n, броят на обхващащите дървета е равно на n^{n-2}. От друга страна от описаният алгоритъм се вижда че при стъпки СТ1 и СТ2 от него се извършват операции с голям брой претърсване и заделяне на изчислителни ресурси. Но за ускоряването на този процес, което е съществено важно при нарастването на n, може допълнително преди стартирането на основния алгоритъм да бъдат извършени някои предварителни стъпки с цел съкращаването на процеса на претърсване. В най общ план се извършва такова пренумериране на върховете на графа G, така че с най малки номера да бъдат “изтеглени” върховете с най-висока степен d*. Т.е. на връх с най-малък номер съответната степен е най-голяма. Друг начин е възможен след предварително пренареждане на ребрата в посока на нарастването на тяхното тегло. Изборът на един или друг вариант на алгоритъма зависи от конкретната конфигурация и свързаността на графа, т.е. съществуват доказателства за обема на операциите в изчислителния процес в зависимост от броя и ребрата, които са посочени особено детайлно в [7], [8]. Така, ако имаме граф G с n на брой върхове и m на брой ребра, за споменатите по-горе два метода съществуват следните изчислителни оценка и за сложността на алгоритъма: $O = м^3/3$ в случай на използване на алгоритъм базиращ се на сравнения на теглата на ребрата, инцидентни с избран отначало връх; $O = м \log\log n$ в случай на използване на алгоритъм, базиращ се на избор на ребра от множеството ребра, сортирани възходящо спрямо тяхното тегло.

Предлаганият математически модел и метод за решаването на такъв тип проблеми в областта на комуникациите е видно , че може да бъде приложен и в областта на отделна банкова компания, където освен със средствата на информатиката за завишаване на сигурността на предаваната информация между отделните звена се създават модели от този род, които със средствата на математическото моделиране и изследване на операциите могат да бъдат решени поставените от тях задачи.

Подобен е и проблема за конфиденционалност и достъп на информация, както и нейното опазване от вредни въздействия в различните организации за сигурност, както и от друга страна това е един начин за оценка на “чистотата” и степента на

риска от “разсейване” или “увреждане” на информацията, взаимодействие и боравене с информация в големи екипи от хора и т.н.

ЛИТЕРАТУРА

[1] Bollobas B.. Modern graph theory. Graduate texts in Mathematics, vol 184, 1998.

[2] Dijkstra E.W., Anote two problems in connection with graphs. Numeric mathematik, 1959,1, p269

[3] Белобаш Б.. Теория на графите."Наука и изкуство", 1990.

[4] Берж К., Теория графов, М., 1962.

[5] Гери М., Д. Джонсон. Вычислительные машины и труднорешаемые задачи.Издательство "Мир", 1982.

[6] Пападимитриу Х., К. Стайглиц. Комбинаторная оптимизация.Алгоритмы и сложнжсть.Издательство "Мир", 1985.

[7] Рейнгольд Э., Нивергельд Ю., Дэо Н. Комбинаторные алгоритмы. Теория и практика. Мир, 1980.

[8] Свами М., К Тхуласираман. Графы, сети и алгоритмы.Издательство "Мир", 1984.

За контакти:

Н.с. Танчо Яков, ИМИ-БАН, секция изследване на операциите, Пловдив, 4000, бул. Руски 139, тел.: 032/625008: tan0702@abv,bg

Докладът е рецензиран.