UNIVERZITA PARDUBICE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Výukový program na řízení rozsáhlých projektů

Bc. Martin Šváha

Diplomová práce

2009

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Katedra softwarových technologií
Akademický rok: **2008/2009**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Martin ŠVÁHA**

Studijní program: **N2646 Informační technologie**

Studijní obor: **Informační technologie**

Název tématu: **Výukový program na řízení rozsáhlých projektů**

Zásady pro vypracování:

Cílem diplomové práce je vytvoření výukové počítačové podpory pro řízení rozsáhlých projektů. Práce bude vycházet z hlubšího poznání problematiky získané studiem doporučené literatury z oblasti síťové analýzy a jejich aplikací v reálných situacích. Dále bude práce vycházet z prostudování řady profesionálních nástrojů běžně dostupných na trhu SW. Navržený model výukového programu bude koncipován tak, aby reflektoval stochastické i determinictické chování doby trvání činnosti. Zabudovaná nápověda umožní uživateli snadnou a rychlou orientaci při řešení rozhodovacích situací v oblasti sestavy síťového grafu projektů, evaluace hran a uzlů grafu a použité metody výpočtu. Model bude v průběhu řešení vyžadovat vícenásobnou kvalifikovanou interakci obsluhy, jejíž správnost bude systémem kontrolována. Efektivnost navrženého programu jak po stránce pedagogické, tak po stránce praktického využití bude diplomantem testována a vyhodnocena.
Struktura DP:
a) Obsah práce
b) Úvod, motivace
c)Historie řešení problematiky řízení projektů
d) Teoretické základy
e) Porovnání vybraných metod pro řízení projektů
f) Formulace problému diplomové práce
g) Návrh modelu počítačové podpory
h) Implementace softwarového řešení
i) Testování efektvnoti navrženého SW
j) Zhodnocení vlastního příspěvku a závěr

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **LEE, S. M., OLSON, D. L. Introduction to management science. Mason, Thomson, 2006.**
2. **NEČAS, J. Grafy a jejich použití. Praha, Státní nakladatelství technické literatury, 1978.**
3. **KLUSOŇ, V. Kritická cesta a PERT v řídící praxi. Praha, Státní nakladatelství technické literatury, 1973.**
4. **SEDLÁČEK, J. Kombinatorika v teorii a praxi. Praha, Československá akademie věd, 1964.**
5. **MAKEOWER, M. S., WILLIAMSON, E. Teach Yourself Operational Research, Problems, Technique and Excercises. Hodder Arnold H&S, 1985.**

Vedoucí diplomové práce: **doc. Ing. Josef Volek, CSc.**
Katedra informatiky v dopravě

Datum zadání diplomové práce: **31. října 2008**

Termín odevzdání diplomové práce: **22. května 2009**

Univerzita Pardubice Česká republika

L.S.

doc. Ing. Simeon Karamazov, Dr.
děkan

doc. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 4. listopadu 2008

PROHLÁŠENÍ AUTORA

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně Univerzity Pardubice.

V Pardubicích dne 22. 05. 2009

Martin Šváha

PODĚKOVÁNÍ

Rád bych touto cestou poděkoval vedoucímu mé práce doc. Ing. Josefu Volkovi za jeho věcné připomínky, rady a konzultace při tvorbě této práce.

Také bych rád touto cestou poděkoval svým rodičům za podporu a poskytnutí zázemí při tvorbě této práce.

ANOTACE

Práce se zabývá problematikou řízení projektů. Vychází z oblasti síťové analýzy a zaměřuje se zejména na metody pro řízení projektů s omezenými zdroji. Porovnává profesionální softwarové produkty dostupné na trhu a popisuje implementaci softwarového řešení.

KLÍČOVÁ SLOVA

Řízení projektů; síťová analýza; grafy; NP-těžké problémy; metoda větví a mezí

TITLE

Educational Software for Large Projects Management

ANNOTATION

This work deals with problems of large-scale projects control. Appears from the network analysis and focuses in particular on methods for the project management with limited resources. The work compares professional software products available on the market and describes the implementation of software solutions.

KEYWORDS

Projects Control; Network Analysis; Charts; NP-hard Problems, Branch and Bound

Obsah

Seznam ilustrací

Seznam tabulek

Seznam zkratek

CPM	Critical Path Method	Metoda kritické cesty
BB	Branch and Bound	Metoda větví a mezí
MS	Microsoft Corporation	Společnost Microsoft
UML	Unified Modeling Language	Unifikovaný jazyk pro tvorbu diagramů
ASME	American Society of Mechanical Engineers	Americké sdružení strojních inženýrů

1 Úvod

Rozvoj výpočetní techniky, který začal v 80. letech minulého století, umožnil projektovému plánování masivní rozšíření téměř do všech hospodářských oblastí. Všechny společnosti, ať už malé či velké, v přítomnosti nebo v minulosti, byly závislé na vzniku a rozvoji nových firem v různých sférách. Tyto podniky se lišily v rozsahu a vývoji nabízených služeb, výrobním procesu, vztahu k zákazníkům nebo politice budování podniku. Zatímco v 80. a 90. letech byl hlavní důraz kladen zejména na kvalitu, ve 21. století tento trend nabral směr k rychlosti. Rychlost je dnes jedním z nejdůležitějších aspektů, které drží podnik vpředu před konkurencí. Společnosti se neustále potýkají s problémy ve vývoji složitých výrobků, služeb a procesů. Mnohdy jsou tyto činnosti spojeny s rychlým dodáním produktu na trh a případně potřebnou odbornou kvalifikací nutnou pro jeho obsluhu, respektive zaškolení. V této situaci se projektové řízení stává velmi důležitým a silným nástrojem v rukou kterékoliv společnosti, které plně chápou význam jeho využívání a dokážou jej aplikovat v praxi.

Možnosti v oblasti řízení projektů umožňují spolu s aplikací informačních řídících systémů vytvořit efektivně týmovou spolupráci, nejen lokálně v rámci firmy, ale i v oblasti partnerských vztahů. Těmito schopnostmi dokáže firma pomocí projektového řízení lépe definovat plány, řídit dodání produktu na trh pomocí synchronizace jednotlivých firemních oddělení nebo kontrolovat velké množství úkolů, časových plánů a alokací zdrojů. To však není vše, čím informační řídící systémy v současnosti disponují. V důsledku synchronizace pracovních metod v reálném čase mohou uživatelé systému současně sledovat nebo pracovat se stejnými informacemi projektu, ať už se jedná o výrobní plány, diskuze, výzkumnou činnost, či jiné relevantní dokumenty.

Na nejvyšší úrovni organizace jsou prováděny řídící techniky, které zajišťují podnikům pružně reagovat v čase na řízení rozpočtu, cen na trhu, komunikace se zákazníky, až po stanovení kvality. Na nejnižší úrovni se používá projektové řízení v kombinaci s informačním systémem. Tato kombinace má za úkol snížení režijních nákladů projektu, vytvořit pracovní prostředí projektu respektující spolupráci jednotlivých týmů v rámci lokálního prostředí firmy, informovat výkonné vrstvy podniku o strategiích v reálném čase, umožnit sdílení účelných informací mezi uživateli a za-

jistit dodržení daných termínů pro dokončení projektu.

Hlavní roli při rozhodování mají stále manažeři a projektanti. Nicméně informační systémy spolu s podporou řízení projektů jim dávají spolehlivý podklad pro rychlejší a přesnější práci. Díky tomu se na trhu objevuje poměrně velké množství softwarových produktů. Tyto nástroje jsou v dnešní době díky rozsáhlé škále algoritmů a způsobů náhledů na problém velmi dobře vypovídající. Takovéto aplikace mají spoustu různých funkcionalit a je pouze na rozhodnutí vedení firmy, který z těchto nástrojů zakoupit s ohledem na cenu a množství využitelných funkcí. Většina softwarových podpor má zaměření pouze pro výsledné řešení, ale již neumožňuje v dostatečné míře pochopení základních principů algoritmů a průběhu výpočtu ukazatelů. Pro používání těchto nástrojů by měl každý uživatel mít základní znalosti této problematiky.

2 Teoretické základy

2.1 Vybrané pojmy z teorie grafů

Incidence p grafu přiřazuje každé jeho hraně neuspořádanou dvojici vrcholů. Je-li incidence hrany $h \in X : p(h) = (u,v)$ hovoříme, že hrana h inciduje s vrcholy u a v. Vrcholy u a v nazýváme krajními body hrany $h \in X$ [1].

Stupeň vrcholu $v \in V$ je počet incidujících hran s tímto vrcholem a značíme jej $st(v)$ [1].

Neorientovaným grafem rozumíme uspořádanou trojici $G = (V, X, p)$. Prvky množiny V nazýváme vrcholy grafu G, prvky množiny X hranami grafu G a zobrazení množiny X do množiny všech neuspořádaných dvojic $V \boxtimes V - p$ incidencí grafu G [1].

Orientovaným grafem $D = (V, Y, p)$ nazýváme uspořádanou trojici: množiny V, Y a zobrazení p množiny $Y \to V \times V$. Množiny V a Y mají stejný význam jako u neorientovaných grafů s tím rozdílem, že množina Y je tvořena uspořádanými dvojicemi $[u, v]$ prvků množiny V, přičemž $u \neq v$ a každá takováto dvojice se vyskytuje v množině Y nejvýše jednou [1].

Vrcholově (hranově) ohodnoceným grafem nazveme graf $G = (V, Y, p)$, pokud existuje funkce $o(v)$ (resp. $o(h)$), která přiřadí každému vrcholu $v \in V$ (hraně $h \in Y$) nezáporné číslo vyjadřující určitou kvantitativní nebo kvalitativní vlastnost vrcholu (hrany) [1].

Sled S je střídavá posloupnost bezprostředně po sobě následujících vrcholů a hran grafu $G = (V, X, p)$, pokud [1]:

$$S = \{u_0, h_1, u_1, h_2, u_2, \ldots, u_{n-1}, h_n, u_n\},$$

$$\text{kde } h_i \in X, p(h_i) = (u_{i-1}, u_i) \text{ pro } i = 1, \ldots, n,$$

$$u_i \in V \text{ pro } i = 1, \ldots, n,$$

$$u_0 = u, u_n = v.$$

Tah je sled, ve kterém se neopakuje žádná hrana [1].

Cesta je tah, ve kterém se neopakuje žádný vrchol. Cestu mezi dvěma vrcholy označujeme $m(u, v)$ [1].

Dráhou (orientovanou cestou) nazveme orientovaný sled, ve kterém se neopakuje žádný vrchol. Orientovanou cestu mezi dvojicí vrcholů *u*, *v* v orientovaném grafu $G = (V, Y, p)$ značíme *m*[*u*, *v*] [1].

Délka cesty mezi dvěma vrcholy $u, v \in V$ hranově ohodnoceného neorientovaném grafu *G* je definována pomocí vztahu [1]:

$$|m(u,v)| = \sum_{h \in m(u,v)} o(h).$$

Vzdálenost dvou uzlů $u, v \in V$ v neorientovaném grafu *G* = (*V*, *X*, *p*) je definována jako:

$$d(u,v) = \min_{m(u,v) \in M} \left\{ \sum_{h \in m(u,v)} o(h) \right\},$$

kde $o(h)$ je ohodnocení hrany $h \in X$ vyjadřující její délku a *M* je množina všech cest mezi vrcholy *u* a *v* [1].

Maximální dráhou z vrcholu *u* do vrcholu *v* orientovaného grafu *G* = (*V*, *Y*, *p*) nazveme dráhu $m^*[u, v]$, pro kterou platí:

$$\sum_{h \in m^*[u,v]} o[h] = \max_{m[u,v] \in M} \left\{ \sum_{h \in m[u,v]} o[h] \right\},$$

kde *M* je množina všech drah *m*[*u*, *v*] [1].

Acyklický graf je orientovaný graf, který neobsahuje žádný cyklus.

Uzel $v_i \in V$ je jedním ze základních prvků síťového grafu. Uzel představuje okamžik v čase, ve kterém dochází k zahájení nebo ukončení činnosti [2].

Činnost je reprezentována hranou $h \in Y$. Jedná se o aktivitu, která je časové ohodnocena. Zpravidla spotřebovává čas, zdroje (materiály, finance) a vyžaduje i lidskou aktivitu.

Síťový graf je orientovaný, neorientovaně souvislý, hranově ohodnocený, acyklický graf, vyjadřující časovou a věcnou návaznost jednotlivých činností projektu [1].

Kritická cesta je libovolná dráha, jejíž součet ohodnocení hran t_n je maximální. Všechny činnosti na této dráze jsou **kritické činnosti**. Prodloužením libovol-

né kritické činnosti o e dojde k prodloužení celkové doby pro dokončení projektu $t_n = t_n + e$ [1].

Matice přímých vzdáleností

Matice přímých vzdáleností hranově ohodnoceného grafu G (V, X, p) je čtvercová matice $D = \left(d_{ij}\right)_{i,j=1}^{n}$, pro kterou platí [1]:

$d_{ij} = o(h)$, jestliže $\exists\ h \in X$, pro kterou $p(h) = (v_i, v_j)$, $i \neq j$

$d_{ij} = \infty$, jestliže $\nexists\, h \in X$, pro kterou $p(h) = (v_i, v_j)$, $i \neq j$

$d_{ij} = 0$, pro $i = j$.

Topologické uspořádání vrcholů a hran grafu G je taková posloupnost $v_1, v_2, \ldots, v_n$ všech vrcholů grafu, že kdykoliv vede orientovaná hrana z v_i do v_j, platí $i < j$. To znamená, že počáteční vrchol hrany je v topologickém uspořádání uveden vždy dříve než koncový vrchol téže hrany. Pokud existuje topologické uspořádání vrcholů a hran, pak je graf acyklický [3].

2.2 Pojmy z teorie složitosti

Výpočetní složitost určuje čas, který je potřeba k proběhnutí algoritmu. Často se určuje podle počtu operací nutných k provedení. Úloha má **horní odhad složitosti** $f(n)$, pokud existuje algoritmus, který dokáže danou úlohu vyřešit v časové složitosti $O(f(n))$ [4].

Asymptotická složitost klasifikuje počítačové algoritmy. Jedná se o funkci udávající jakým způsobem se bude chování algoritmu měnit v závislosti na změně velikosti vstupních dat [4].

Polynomiální výpočetní složitost je složitost, kterou lze popsat funkcí $O(p(n))$, kde $p(n)$ je polynom. Algoritmus má polynomiální složitost, pokud jej lze řešit v polynomiálním čase [4].

Nepolynomiální výpočetní složitost je taková složitost, u které nelze určit horní odhad složitosti. Znamená to, že nedokážeme výpočetní náročnost dané funkce shora ohraničit polynomiální funkcí [4].

Třída P je třída úloh, které jsou řešitelné deterministickým Turingovým strojem v polynomiálním čase. Tj. řeší úlohu, která má složitost $O(p(n))$, kde $p(n)$ je po-

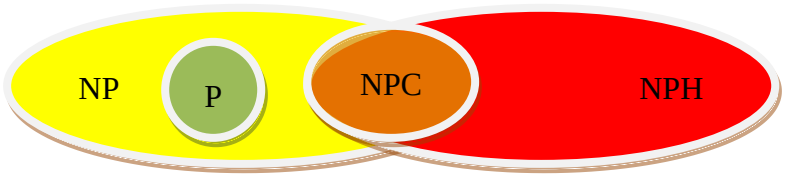
lynom [4].

Třída NP je třída úloh, které jsou řešitelné pomocí nedeterministického Turingova stroje v polynomiálním čase. Obecně lze říci, že pro danou úlohu nejsme schopni nalézt řešení v polynomiálním čase, ale pouze toto řešení přibližně ověřit. V principu nedeterministický Turingův stroj v každém kroku může výpočet rozvětvit do *n* větví [4].

Třída NPC neboli NP-úplná je speciálním případem NP úloh. Pro všechny úlohy z množiny NP platí, že lze úlohu redukovat (převést) na úlohu NP-úplnou. Úloha *A* je polynomiálně redukovatelná (převoditelná) na *B*, pokud existuje takové zadání *B*, že z výsledku řešení *B* je možné odvodit řešení *A*. Je-li *B* polynomiálně řešitelná úloha a *A* je polynomiálně převoditelná na *B*, pak i *A* je polynomiálně řešitelná [4].

Třída NPH neboli NP-těžká je třída úloh, na kterou se polynomiálně redukuje každá NP úloha [4].

Množina všech P úloh je podmnožinou NP úloh. Není dokázáno, ani se nepředpokládá, že tyto množiny jsou totožné. Třída NPH obsahuje třídu NPC a zároveň úlohy NPC jsou podmnožinou NP.



Obrázek 1 - Třídy složitosti – Zdroj: [4]

3 Historické pozadí řízení projektů

3.1 Uvedení do historie řízení projektů

Mohly být křížové výpravy započaty, vojáci vyzbrojeni a stravováni bez účinného řízení projektu? Mohla by být postavena Velká čínská zeď týmem čítajícím několik milionů lidí a v rozpětí několika tisíců let bez použití řízení projektu? Je možné říci, že koncepce projektového řízení byla hotova zhruba již v počátcích psané historie. Myšlenka plánování není nová. Shun Tzu napsal o plánování a strategii z vojenského pohledu již před 2500 lety. Pyramidy byly postaveny před více jak 4500 lety, Taj Mahal v 17. století, či transkontinentální železnice ve Spojených státech amerických před více jak 200 roky. Tyto a mnohé další projekty potřebovaly určitou míru plánování.

Ačkoliv existence plánování sahá do hluboké minulosti, o plánování jako vědecké disciplíně se začalo mluvit až na začátku druhé poloviny 20. století. Teprve až projekt Manhattan ve 40. letech minulého století určil rozdílné role vojenských úředníků a vědců pro plánování projektů.

Během průmyslové revoluce došlo k růstu v oblastech průmyslu a podnikání, které se prudce rozšířily po všech kontinentech. S nástupem automatizace, vše začalo být prováděno v mnohem větším měřítku. Schopnost řídit projektové rozpočty, dodávky materiálů, koordinace práce na různých místech byla zcela zásadní pro vznik potřeby zkoumání nových myšlenek pro zefektivnění metod řízení.

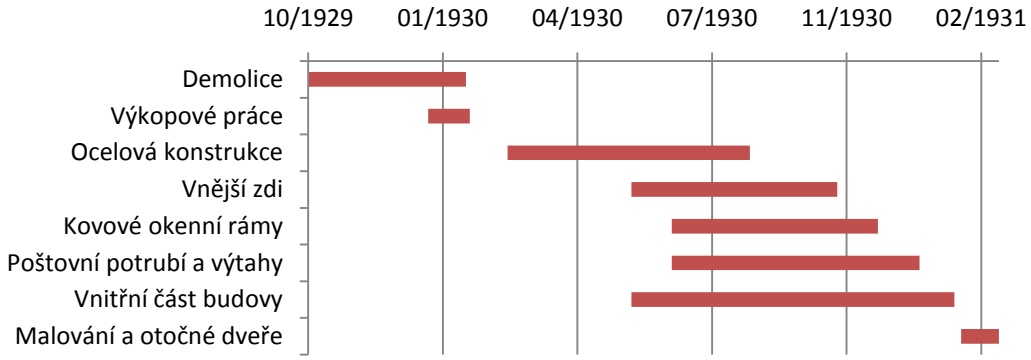
3.2 Počátek řízení projektů jako vědecké disciplíny

Za průkopníka vědeckého řízení projektů je považován Frederick Winslow Taylor. Na počátku 19. století ve Spojených státech amerických zkoušel své teorie na zvýšení produktivity vytvořením metodiky pro měření a plnění některých úkolů pracovníků v ocelárnách. Zajímal se o objevování nových a lepších způsobů, jak by zaměstnanci mohli zlepšit pracovní výkon, aniž by pracovali tvrději a déle. Jeho hlavními myšlenkami bylo za pomocí vědy vybrat, trénovat a rozvíjet každého zaměstnance samostatně a neponechat je jejich pasivnímu individuálnímu tréninku. Další ideou bylo poskytovat podrobné instrukce a dohlížet na všechny zaměstnance při jejich pracovním výkonu. To znamenalo vytvořit pro každého pracovníka individuální pokyny. A v neposlední řadě měla být práce rovnoměrně rozdělena mezi manažery a zaměstnance, tak aby manažeři uplatňovali zásady vědeckého řízení pro

plánování práce a zaměstnanci skutečně plnili úkoly. Frederick Taylor zemřel roku 1915 ve Philadelphii a nápis na jeho náhrobním kameni určil pevně jeho místo v historii - "otec vědeckého řízení projektů." [5]

3.3 První grafická zobrazení pro plánování

První diagramy pro plánování mají své kořeny již v roce 1765. Angličan Joseph Priestly je popsal jako sloupcové diagramy ve své práci „Chart of Biography“. Tento koncept využil Taylorův přítel Henry Laurence Gantt, který je považován za prvního autora diagramů pro dokumentování, měření a řízení procesů. Tyto diagramy byly již vytvořeny Karolem Adamieckim v roce 1896, který je nazval harmonogramem. Adamiecki ovšem tyto diagramy publikoval až v roce 1931, zatímco Henry Gantt již v roce 1913 ve své knize „Work, Wages, and profits“. Proto jsou dnes označovány jako Ganttovy diagramy. [5]



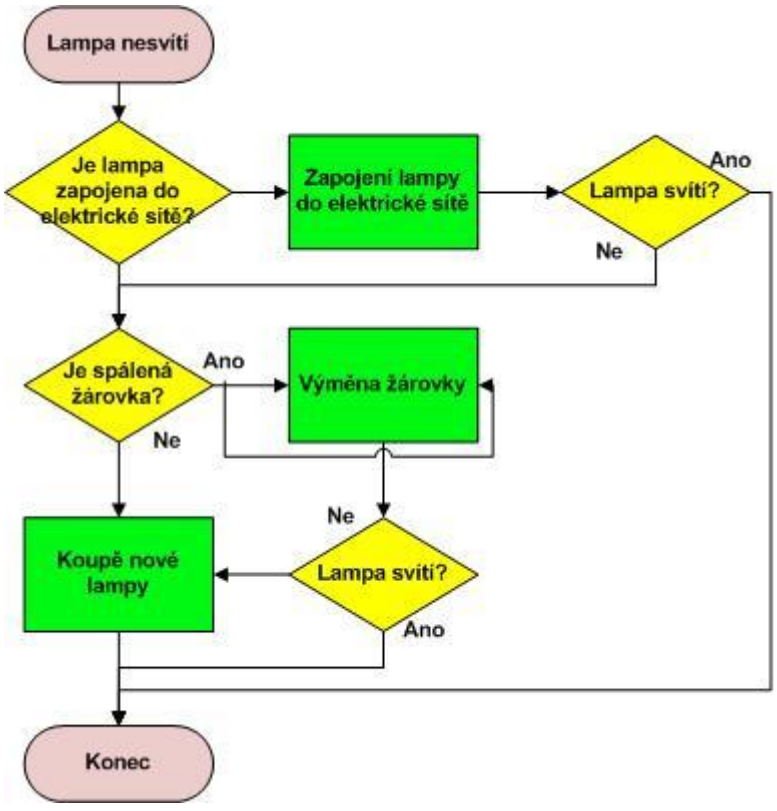
Obrázek 2 - Ganttův diagram stavby Empire State Building - Zdroj[1]

V původní formě diagramy popisovaly pouze časovou následnost, ale nedokázaly určit jednotlivé návaznosti činností mezi sebou. Henry Gantt vytvořil tyto diagramy během první světové války pro stavbu lodi amerického námořnictva. Díky zmapování a analýze jednotlivých kroků budování lodi byl schopen sledovat celkový obraz projektu a měl k dispozici výpis informací o vztazích mezi činnostmi. Ganttovy diagramy se staly důležitým nástrojem pro řízení projektů a jsou používány již přes 100 let. Příkladem projektu realizovaného s využitím Ganttových diagramů je stavba Empire State Building 1929-1931, či výstavba Hoover Dam v letech 1931-1936, která zaměstnala přibližně 5200 pracovníků a je stále jednou z nejvyšších pře-

[1] HUML, Don. *Don Huml's view on Operations* [online]. 2009 [cit. 2009-05-07]. Dostupný z WWW: <http://donhuml.com/operations.html>.

hrad v USA vyrábějící přibližně čtyři miliardy kilowatthodin ročně. Dalším příkladem využití byl projekt Manhattan v letech 1942-1945, který byl prvním v oblasti výzkumu a vývoje výroby nukleárních zbraní. Na projektu pracovalo cca 125.000 pracovníků a náklady dosáhly téměř 2 miliard dolarů. [5]

Objevily se i další metody řízení projektů, které danou problematiku graficky zobrazovaly odlišným způsobem. Jednou z nich byly tzv. postupové (v některých literaturách označovány jako vývojové) diagramy, které umožnily vyloučit zbytečné pohyby na pracovišti, zvýšit pracovní výkon a realizovat projekt v nejkratším možném čase s minimálním pracovním úsilím. Postupový diagram je obecný typ grafu sloužícího k zobrazení algoritmů nebo procesů. První postupový diagram byl představen v roce 1921 Frankem Gilbrethem před členy ASME (American Society of Mechanical Engineers) jako prezentace „Process Charts-First Steps in Finding the One Best Way“. Gilbrethovy nástroje se velmi rychle rozšířily v průmyslové sféře. Allan Mogensen začal tyto metody začátkem 30. let přednášet ve vesnici Lake Placid ve státě New York. V roce 1944 absolvent Mogensenovy třídy, Art Spinanger, na základě těchto metod vytvořil projekt “Deliberate Methods Change” ve firmě Procter and Gamble, který dle zprávy z října roku 1984 ušetřil 900 miliónů dolarů. [5]



Obrázek 3 - Postupový diagram opravy lampy – Zdroj: vlastní

Již před první světovou válkou Společnost německých inženýrů REFA a němečtí podnikatelé se snažili o aplikaci Taylorova konceptu do svých podniků. Němečtí inženýři založili na vlastních výzkumech systém REFA – systém normování pracovního času, který pomohl vyvést Německo z ekonomické krize po 1. světové válce. Název REFA pochází ze slovního spojení „Reichsausschuss für Arbeitszeitermittlung“. Metodika systému REFA vznikla na základě Taylorových a Gilbrethových prací o časových a pohybových studiích. Tento systém znázorňuje pracovní činnosti. Sytém REFA zahrnuje i grafické znázornění, které lze považovat za předchůdce síťových grafů. REFA rozebírá příliš složité činnosti, dokud každá složka není jasná a časově měřitelná. Celkový čas projektu je pak vyjádřen součtem časů dílčích činností projektu. V současné době má REFA Association přes 18,000 členů. Mezi nimi jsou i přední světový výrobci jako Volkswagen, Adidas, Siemens a další [5] [7].

Rozvoj průmyslových odvětví a různých vědeckých oborů vedl k potřebě plánování, koordinace a kontroly rozsáhlých a složitých projektů v těchto oblastech. Pro řízení úkolů byla vytvořena síťová analýza, což je souhrnný název pro metody, jejichž základ tvoří teorie grafů, teorie pravděpodobnosti a vědecké programování. Základ síťové analýzy tvoří metoda kritické cesty (CPM – Critical Path Method) a plánovací systém PERT (Program Evalution and Review Technique). Dále do této oblasti spadá nespočet modifikací těchto metod. Obě základní metody se vyznačují svou jednoduchostí, přehledností a snadnou řešitelností.

3.4 Vytvoření metody kritické cesty

V polovině roku 1956 společnost E. I. Du Pont de Nemours & Co. hledala užitečný způsob využití jejich počítače UNIVAC1 (byl to první počítač, který byl použit pro obchodní účely). Vedení Du Pont cítilo velkou příležitost využít výpočetní techniky pro plánování, předběžné výpočty a postupy. Cílem tohoto vývoje bylo vytvoření účinného nástroje pro řízení složitých činností ve výstavbě výrobních zařízení, rekonstrukce zařízení, v oblasti údržby a při vývoji nových chemických výrobků. Touto úlohou se začal zabývat zaměstnanec firmy Morgan Walker. Spolu s J. E. Kellym měli vyřešit hlavolam časové náročnosti projektu. Zaměřili se, jak získat zpět ztracený čas u činností, u kterých došlo k překročení požadované doby trvání. Snahou bylo nalézt činnosti, u kterých by bylo možné snížit čas trvání bez navýšení ná-

kladů. 7. května 1957 uvolnila společnost Du Pont 228000 amerických dolarů na vývoj vhodné metody. Klíčovými členy řešitelského týmu byli Morgan Walker, expert ze společnosti RAND James E. Kelley a John Mauchly. Kellyho řešení bylo založeno na lineárním programování a používalo i-j notace určující vztah mezi jednotlivými činnostmi. Sběr dat pro CPM si vyžádal přes tři měsíce. Za oblast sběru dat odpovídal M. Walker. 24. července 1957 byla Georgem Fischerem vytvořena první analýza a byl ověřen celkový princip. Fisherův plán obsahoval 61 činností, 8 z nich bylo kritických a 16 fiktivních [6].

Téhož roku byla společnostmi E. I. Du Pont de Nemours & Co. a Sperry-Rand Corporation publikována metoda kritické cesty. První zkušební aplikace se uskutečnila v září 1957. Její výsledky byly dokončeny v květnu následujícího roku. Jednalo se o výstavbu zařízení v hodnotě 10 mil. dolarů. Ukázalo se, že lze přesněji stanovit počet pracovníků a zároveň zkrátit dobu realizace o 2 měsíce a to bez zvýšení nákladů. Při vynaložení nákladů vyšších o 1% bylo možné zkrátit dobu o další 2 měsíce. Tyto výsledky vzbudily v odborné veřejnosti velkou pozornost. Další oblastí využití aplikace metody CPM byla oblast údržby. Šlo o preventivní údržbu reaktoru v továrně na výrobu neoprénu v Louisville. Aplikace byla ukončena v roce 1959 a její výsledky byly překvapující. Podařilo se snížit dobu trvání ze 125 hodin na 93. Bylo nalezeno 25 kritických činností (z celkových 225), jejich urychlením bylo možné snížit dobu trvání opravy až na 78 hodin. Významné změny byly provedeny i v oblasti postupu práce [2].

3.5 Vývoj metody PERT

Nezávisle na vývoji metody CPM tým operační analýzy v Úřadu válečného námořnictva USA roku 1957 začal vyvíjet nové řídící metody v oblasti vývoje raketových systémů. Jednalo se o vývoj systému raketových nosičů balistických střel Polaris. Na vývoji spolupracovali Williard Fazar, pracovníci společnosti Booz Allen & Hamilton, zástupci společnosti Lockheed Missiles a pracovníci Úřadu válečného námořnictva. V červnu roku 1958 vydala skupina první zprávu "PERT, Summary Report, Phase I". V říjnu tohoto roku byl navržený systém zkušebně aplikován pro vývoj tří základních tří vyvíjené zbraně. Výpočty proběhly na IBM Naval Ordinance Research Computer (NORC) v Dahlgrenu ve Virginii. Šlo o 23 síťových diagramů, 2000 uzlů a 3000 činností. Uvádí se, že využitím metody PERT došlo ke zkrácení

vývoje rakety Polaris o dva roky. Za autory jsou považováni D. G. Malcolm, J. H. Roseboom, W. Fazar a C. E. Clark, který položil důležité základy ve své publikaci o síťových grafech. Obě metody CPM a PERT se rychle rozšířily zejména ve stavebnictví a průmyslu [6].

V 60. letech vzniklo několik modifikací metody PERT. NASA vyvinula nebo přepracovala několik nástrojů, jako například WBS (Work Breakdown Structure), PERT/Cost, PERT-RAMPS (Resource Allocation & Multi-Project Scheduling), zaměřených zejména na kritérium omezení nákladů.

Systém WBS je založen na rozpracované struktuře nebo osnově rozpisu práce. Pomocí WBS je možné hierarchicky rozčlenit projekt do dílčích fází projektu, skupin úkolů, až na nejnižší úroveň struktury projektu. Problémem tohoto systému je, že na vysoké úrovni zobrazení neukazuje závislosti mezi činnostmi ani jejich doby trvání.

Metoda PERT/Cost byla založená na metodě PERT a jejím stochastickým náhledem na dodržení plánovaného rozvrhu činností kombinované s ukazatelem zvýšením nákladů, aby bylo možné dodržet časový plán

Metoda RAMPS byla původně vyvinuta roku 1962 ve společnosti C-E-I-R (Council for Economics and Industrial Research). Základy této metody jsou postaveny opět na obecných principech síťové analýzy. Navíc RAMPS využívá odhad celkového objemu práce a vytíženosti zdrojů [6].

3.6 Algoritmy pro diskrétní a kombinatorickou optimalizaci

Postupem času byly rozšířeny dalšími postupy pro specifické účely. V roce 1960 byla představena metoda větví a mezí (Branch and Bound) pro lineární programování. Autory jsou A. H. Land a A.G. Doig. Základem této metody je průchod prostorem jednotlivých podproblémů, který vylučuje možnosti mající nepříznivou dobu trvání. Úspěch této metody spočíval v její jednoduchosti. Využívána je pro vyhledávání různých optimálních řešení, zejména v diskrétní a kombinatorické optimalizaci [8].

S rozvojem teorie složitosti na počátku 70. let bylo již zřejmé, že většina z těchto kombinatorických problémů je NP-těžkou úlohou a není možné pro ně vytvořit efektivní postupy pro nalezení jejich řešení pomocí exaktních metod jako je

metoda větví a mezí. Začal tak vývoj heuristických metod, tj. metod pro přibližné výpočty. Bylo navrženo mnoho metod pro řešení NP-těžkých problémů, se kterými se dále experimentovalo.

Nejvíce oblíbenými se díky své jednoduchosti staly metody založené na lokálním prohledávání. Základním principem těchto metod bylo kompletní prohledání těsného okolí a určení směru největšího spádu účelové funkce pseudogradientu. Výsledkem bylo nalezení lokálního řešení, které často bylo podprůměrným a tudíž nevyhovovalo daným potřebám. Problémy pasti lokálního optima se snažily vyřešit další metody.

V roce 1983 dostali S. Kirkpatrick, C. D. Gelatt a M. P. Vecchi myšlenku, že hledání optimálního řešení může být realizováno podobným způsobem jako žíhání tuhého tělesa. Tento způsob nazvali metodou simulovaného žíhání a publikovali jej v knize „Optimization by Simulated Annealing". Tuto metodu popsal nezávisle na práci uvedených třech autorů V. Černý v roce 1985 v publikaci „A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm." U metody lokálního prohledávání existoval problém s uváznutím v řešení lokálního minima. Metoda simulovaného žíhání se snažila problém odstranit tím, že na začátku řešení se prováděly velké změny. Tyto změny byly určeny teplotou. Čím vyšší teplota, tím větší změny. U metody lokálního prohledávání byl postup pouze k lepšímu řešení, zatímco algoritmus simulovaného žíhání bral v potaz s určitou pravděpodobností i řešení, která byla horší. V průběhu algoritmu teplota klesala v závislosti na konvergenci algoritmu, aby bylo případně možné dostat se z lokálního řešení [9].

V roce 1986 navázal Frederick Glover na metodu lokálního prohledávání v knize "Future Paths for Integer Programming and Links to Artificial Intelligence" a odstranil nedostatek uváznutí v lokálním minimu. Tuto metodu publikoval pod názvem Tabu Search (metoda zakázaného prohledávání). Ve skutečnosti mnoho prvků tohoto návrhu zakázaného prohledávání bylo publikováno již v roce 1977 (Glover), včetně krátkodobé paměti, která měla umožnit zrušení případných posledních tahů. Základním principem této metody je, aby pokračovala i v případě, kdy narazí na lokální optimum. Pomocí tahů, které nevylepšují řešení, umožňuje metoda vrátit se zpět na již dříve nalezené řešení. Seznam kroků, které si algoritmus pamatoval, se nazývá zakázaný seznam. Je zajímavé poznamenat, že ve stejném roce Hansen

navrhl podobný přístup, který nazval metodou nejprudšího stoupání [10].

V následujících letech vzniklo mnoho dalších nových přístupů, většinou založených na analogii s přírodními živly, spolu s několika staršími metodami, jako jsou genetické algoritmy (Holland, 1975), které získávaly rostoucí popularitu. Souhrnně se začaly tyto metody označovat pod společným názvem meta-heuristické metody. Tyto metody stanuly v posledních letech v čele heuristických přístupů k řešení kombinatorických optimalizačních problémů.

3.7 Počítačová éra

Prvních využití počítačového softwaru pro řízení projektů bylo vyvinuto britskou firmou Micro Planning Services pro počítač Apple II Micro Planner v1.0 v roce 1980. Vývoj systému trval celých 14 měsíců a byl postaven na ICL PERT. V roce 1989 vydala společnost Apple pro počítač Apple Macintosh první graficky zpracovaný software pro řízení projektů Micro Planner X-Pert. Série Micro Planner vydělala mezi lety 1986 až 1998 přes 1 milión amerických dolarů.

V 80. letech vznikaly i další softwarové nástroje. Roku 1983 Dick Faris, Joel Koppelman a Les Seskin založili firmu Primavera, která je dodnes jedna z dominantních společností na trhu s nástroji pro řízení projektů. Dalším důležitým milníkem na softwarovém poli bylo vydání Microsoft Projectu roku 1984 pro DOS, který je dnes jedním z nejvíce používaných nástrojů [6].

4 Základy síťové analýzy

Síťová analýza je souhrnný název pro metody, které mají společný základ v teorii grafů, teorii pravděpodobnosti a vědeckého programování. Její využití se nachází v oblasti plánování, koordinace a kontroly různých složitých úloh, a to zejména v oblasti stavebnictví, průmyslu, výzkumu a vývoje, pracovních postupů apod. Základem síťové analýzy jsou metody CPM (Critical Path Method) a PERT (Program Evaluation and Review Technique). Tyto metody dostály řady modifikací, jelikož se vyznačují svou jednoduchostí v základní podobě. Obě metody jsou velmi účinným nástrojem pro projektové plánování.

Projekt je možné popsat například následujícím způsobem [2]:

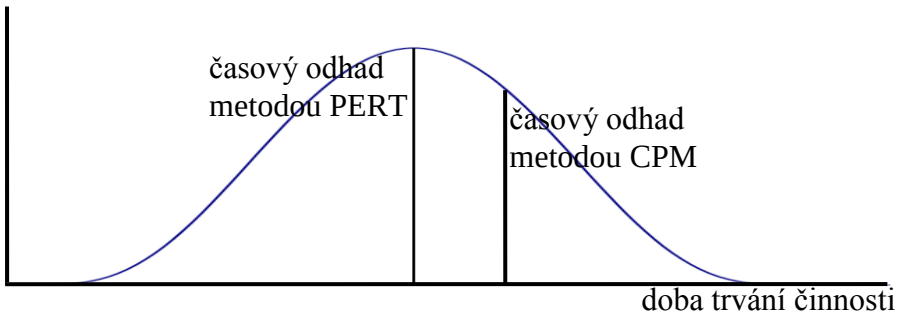
1. Soubor činností, které na sebe určitým způsobem navazují, nebo jsou prováděny souběžně. Činnosti jsou zpravidla vykonávány jednou, tedy jsou většinou neopakovatelné.
2. Soubor činností je konečný a často je výstupem pouze jediný produkt.
3. Dílčí činnosti se chovají jako celek. Samostatně nejsou ve větší míře závislé na vnějších vazbách, které jsou významné spíše pro soubor činností jako celek.
4. Realizace jednotlivých činností jsou spojeny s určitou mírou nejistoty. Může se jednat například o včasné dokončení dílčí činnosti, či navýšení nákladů apod.
5. Jednotlivými úkoly se mohou zabývat různé skupiny. Je tedy kladen velký důraz na koordinaci projektu.

Systém pro řízení projektů by měl splňovat určité podmínky, které nám nad ním umožní efektivní správu. Základními prvky, které by měl systém obsahovat jsou následující [2]:

1. Poskytovat vyčerpávající informace o všech vztazích a závislostech mezi jednotlivými činnostmi takovým způsobem, abychom byli schopni vytvořit přesný přehled pracovního postupu.
2. Možnost efektivní koordinace všech prvků, které se podílejí na daném projektu.

3. Vytvoření lhůtového plánu umožňujícího vyhodnocení jednotlivých činností z hlediska včasného splnění celého projektu, kontrolu postupu práce a případné prognózy pro výskyt některých problémů, aby bylo možné vytvořit opatření, pokud by došlo ke vzniku zpoždění.
4. Umožnit efektivní využití všech prostředků pro realizaci projektu. Například využít maximálně pracovních sil a dalších zdrojů projektu. Dále by měl pomáhat při rozhodování vytvořením různých ukazatelů, například včasné dokončení projektu apod.

Těmito požadavky disponují obě metody, tedy metoda kritické cesty, tak i metoda PERT. Tyto metody se od sebe liší různým pohledem na časový průběh činností. U metody CPM jsou časové náročnosti dílčích činností pevně dány, zpravidla to bývá odhadem. Často nastává problém, že doba trvání činnosti má charakter náhodné veličiny a není možné určit časové ohodnocení činnosti odhadem. Takový případ řeší právě metoda PERT, která ohodnocuje činnosti třemi odhady a s tímto ohodnocením poté pracuje jako s náhodnou veličinou. Na následujícím obrázku je znázorněna doba trvání činnosti náhodnou veličinou a hodnoty doby trvání činnosti reprezentující náhledy metod CPM a PERT. Systém PERT navíc umožňuje při předem stanovených termínech, jako je třeba ukončení celého projektu nebo jeho důležitých etap, určit pravděpodobnost dodržení termínů projektu.



Obrázek 4 - Průběh doby trvání činnosti – Zdroj: vlastní

Pro realizaci projektu mnohdy nestačí znát jen časový průběh jeho dílčích činností, ale je nutné řešit i případný nedostatek zdrojů pro dokončení všech prací. Každá činnost vyžaduje pro svoji realizaci pracovní síly a spotřebovává určité materiály. Tyto prostředky souhrnně nazýváme zdroje. Může nastat situace, kdy potřebných zdroj je nedostatek. Takovéto problémy mohou ohrozit termín dokončení projektu. Zdroje projektu nejsou zpravidla využívány stejnoměrně, ale v některých

obdobích jsou velmi vytíženy, v jiných jejich využití klesá pod průměr. Obě metody řeší tuto problematiku a provádějí rozbor, zda-li je nutné započít práce na jednotlivých činnostech dříve, či je zrychlit nebo zpomalit.

Před vytvořením síťového grafu je důležité popsat časovou organizaci celého projektu, určit závislosti dílčích činností, respektovat kapacity dostupných zdrojů a pracovních sil. Pro sestavení síťového grafu je nutné nejprve určit, jaká metoda síťové analýzy bude použita. Zda se jedná například o metodu CPM nebo PERT, jelikož jednotlivé metody vyžadují odlišná vstupní data a disponují rozdílnými ukazateli charakterizující projekt.

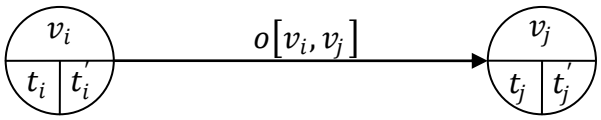
4.1 Sestavení síťového grafu

Základem metod CPM a PERT je grafické vyjádření projektu pomocí síťového grafu. Jednotlivé dílčí činnosti jsou v grafu znázorněny pomocí orientovaných hran a každá z těchto činností je určena dvěma uzlovými body, které se zpravidla značí kruhy, čtverci apod. Každá hrana mezi uzly odpovídá určité činnosti. Takovýto způsob k řešení většiny problémů nestačí. Činnosti mohou být reálné, znázorněné zpravidla plnou čarou, a fiktivní činnosti znázorněné čárkovanou hranou. Reálná činnost představuje postup v realizaci celého projektu vpřed, je nutné na ni vynaložit specifické náklady a vykazuje spotřebu času. U jednotlivých dílčích činností nestačí pouze určit, jakým způsobem činnosti vykonat, ale i v jakém pořadí je vykonat a vyjádřit závislosti, jak na sebe činnosti navazují. Jako prostředek k zachycení složitějších vazeb slouží fiktivní činnosti. Nespotřebovávají žádný čas, ani na ně nejsou vynaloženy žádné náklady. Jedná se tedy pouze o pomocný prvek pro grafické znázornění projektu. Bližší informace o využití fiktivních činnostech je uvedeno v pravidlech pro sestavení síťového grafu.

Dalším základním prvkem pro grafické znázornění projektu je uzel, což je okamžik v čase, ve kterém dochází k zahájení nebo ukončení činnosti. Všechny činnosti vycházející z jednoho vrcholu mají společné všechny bezprostředně předcházející činnosti. Uzly jsou v tomto směru pohledu brány jako jakési kontrolní body projektu, či v některých případech, kdy ohraničují některé důležité fáze projektu, jsou nazývány milníky. Výjimku tvoří počáteční nebo koncový vrchol, do něhož nevchází žádná z hran, respektive z něj žádná hrana nevychází [2].

4.1.1 Základní pravidla pro sestavení síťového grafu:

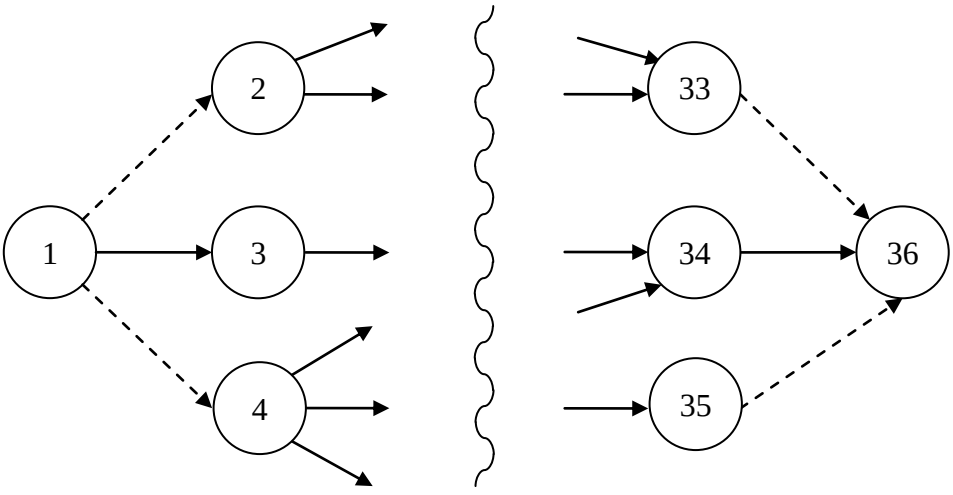
1. Každá činnost má vždy jeden uzel počáteční a jeden uzel koncový. Těmito vrcholy je jednoznačně určena každá činnost grafu. Počáteční uzel se označuje indexem *i* a koncový uzel indexem *j*. Všechny činnosti jsou tedy uspořádanou dvojicí čísel [*i*, *j*].



Obrázek 5 - Znázornění činnosti v síťovém grafu – Zdroj[1]

2. Síťový graf má vždy jeden počáteční a jeden koncový uzel. Pokud existuje více potencionálních počátečních uzlů, pak zvolíme jeden z těchto uzlů počátečním a z tohoto uzlu vedeme ke zbylým počátečním uzlům fiktivní činnosti. Tento proces se nazývá normalizace síťového grafu. Stejným způsobem probíhá i normalizace koncových uzlů. Příklad normalizace síťového grafu je na obrázku 7.

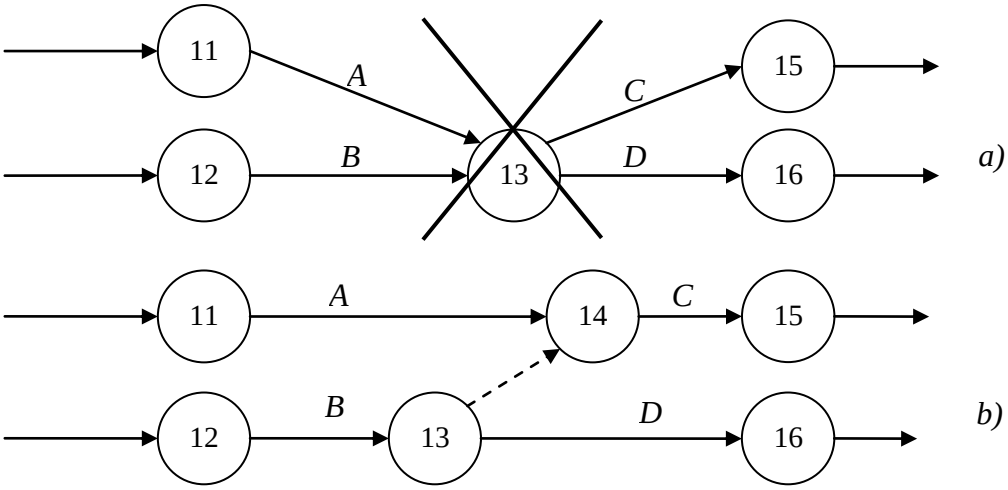
3. Žádná činnost nemůže být vykonána, dokud nejsou dokončeny všechny činnosti jí předcházející. To znamená, že nelze časově dosáhnout činnosti, která nemá ukončené všechny předcházející činnosti.



Obrázek 6 - Normalizace síťového grafu – Zdroj [2]

4. Síťový graf musí korektně popisovat závislosti dílčích činností. Závislé činnosti oddělujeme od činností nezávislých pomocí fiktivních činností. Závislost činností je v uzlech, kde dvě nebo více činností končí, respektive vychází

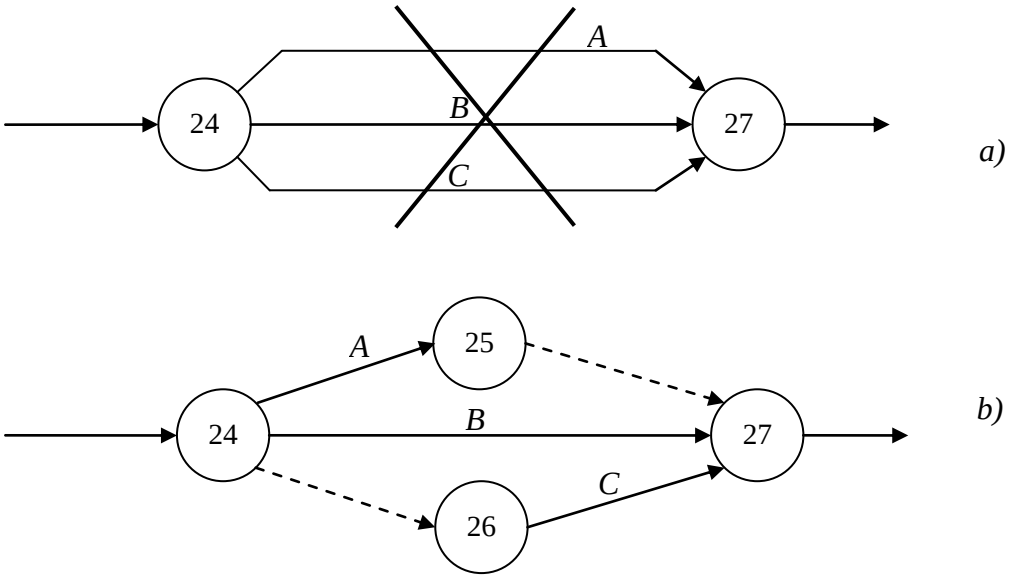
ze společného uzlu. Pokud jen určitá část činností vycházejících z konkrétního uzlu závisí na dokončení předcházejících činností, pak tyto závislé činnosti oddělíme od ostatních, nezávislých, činností pomocí fiktivní činnosti. Jeden z těchto případů je znázorněn na následujícím obrázku.



Obrázek 7a, b - zobrazení nezávislých činností – Zdroj [2]

Činnost *D* závisí na ukončení činnosti *B*, nikoliv však na činnosti *A*. Naproti tomu činnost *C* závisí na dokončení činností *A* i *B*. Díky fiktivní hraně je činnost *D* uvolněna ze závislosti na činnosti *A*. Situace na obrázku 3a by byla správná pouze tehdy, pokud by činnosti *C* i *D* závisela na obou předcházejících činnostech *A* a *B*.

5. Souběžné činnosti v síťovém grafu jsou oddělovány činnostmi fiktivními. Znázornění souběžných činností nesmí být v rozporu s pravidlem jednoznačného určení činnosti dvěma uzly. Proto se opět používají činnosti fiktivní. Nezáleží, zda fiktivní činnost předchází nebo navazuje na činnost souběžnou. Pro *n* souběžných reálných činností platí, že musí být doplněny o (*n - 1*) fiktivních činností. Přiklad třech souběžných činností je na obrázku 4. V tomto obrázku jsou pomocí dvou uzlů *24* a *27* určeny tři činnosti. Proto byly zavedeny fiktivní činnosti (*25, 27*) a (*24, 26*). Přičemž v prvním případě fiktivní činnost následuje za reálnou činností *A*, zatímco v případě druhém fiktivní činnost předchází činnosti *B*. Tedy je zde zobrazeno i pravidlo, že u souběžných činností nezáleží, zda reálným činnostem fiktivní činnost předchází, nebo z ní vychází.



Obrázek 8a, b - zobrazení souběžných činností – Zdroj [2]

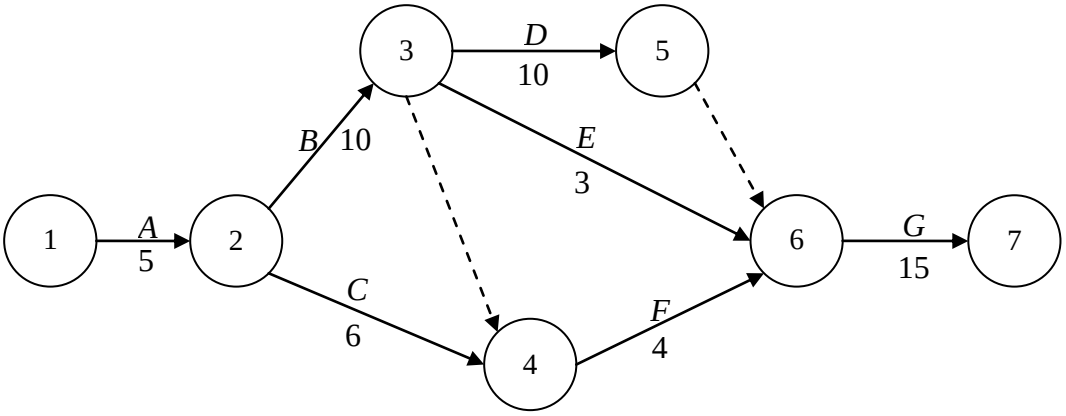
6. Délka hrany zobrazující fiktivní činnost nemá žádný časový význam. Pouze znázorňuje vazby mezi činnostmi.
7. Pokud potřebujeme vyjádřit skutečnost, že započetí některé z činností vyžaduje skončení některých vnějších pracích, či rozhodnutí, pak vytvoříme tzv. předstihovou činnost počátečnímu uzlu. Tato činnost bývá obvykle znázorněna lomenou čarou.
8. Síťový graf nesmí obsahovat cyklus. Vznik v cyklu znamená, že graf obsahuje logickou chybu a nelze tedy projekt nikdy dokončit.

Příklad síťového grafu je na obrázku 5, kde je uvedena výstavba městského domu, skládající se z 9 činností, z nichž jsou 2 fiktivní. Projekty jsou většinou zadávány pomocí přehledné tabulky. Následující tabulka 1 má v prvním sloupci název aktivity, respektive její zkratku pro popis v síťovém grafu, v dalších sloupcích jsou uzly určující činnost, popis činnosti, předchůdci činnosti a v posledním sloupci časové trvání činnosti ve dnech.

Činnost	Uzly	Popis	Předchůdce	Doba trvání
A	1-2	Základové práce	-	5
B	2-3	Stavba zdí a střechy	A	10
C	2-4	Podlahové práce	A	6
D	3-5	Úprava terénu	B	10

E	3-6	Zavedení elektřiny	B	3
F	4-6	Instalatérské práce	B, C	4
G	6-7	Dokončovací práce	D, E, F	15

Tabulka 1 - Stavba městského domu – Zdroj: [14]



Obrázek 9 - Síťový graf stavby městského domu – Zdroj: [14]

4.2 Ganttovy diagramy

4.2.1 Popis Ganttova diagramu

Ganttovy diagramy slouží zejména pro zobrazení časové náročnosti celého projektu, či jeho jednotlivých činností. Do popředí zde vystupuje vztah mezi operacemi a časem. Tyto diagramy jsou kresleny ve specifickém časovém měřítku. Díky tomuto měřítku se jednotlivé činnosti zobrazují jako úsečky úměrné délce jejich časového trvání. V důsledku těchto principů se využívají Ganttovy diagramy jako nástroj pro sestavení časového plánu nebo i třeba pro kontrolu postupu prací projektu. Grafické znázornění Ganttova diagramu jednoduchým způsobem výstižně popisuje znázornění projektu, a to je důvodem, proč je masově rozšířen v řídící a plánovací praxi.

Síťová analýza dále propracovává hlavní myšlenku Ganttova diagramu. Zejména oproti Ganttovým diagramům klade důraz ve zobrazení závislostí mezi činnostmi. Tento důsledek vede k detailnějšímu popisu jednotlivých činností v síťovém grafu. Na druhou stranu síťové grafy nezohledňují délku úseček, zobrazující činnosti, vzhledem k délce trvání činnosti. Tedy tento grafický způsob nelze zohlednit pro zachycení skutečného postupu vykonané práce.

Z předchozího popisu vyplývá, že pomocí Ganttova diagramu nejsme schopni

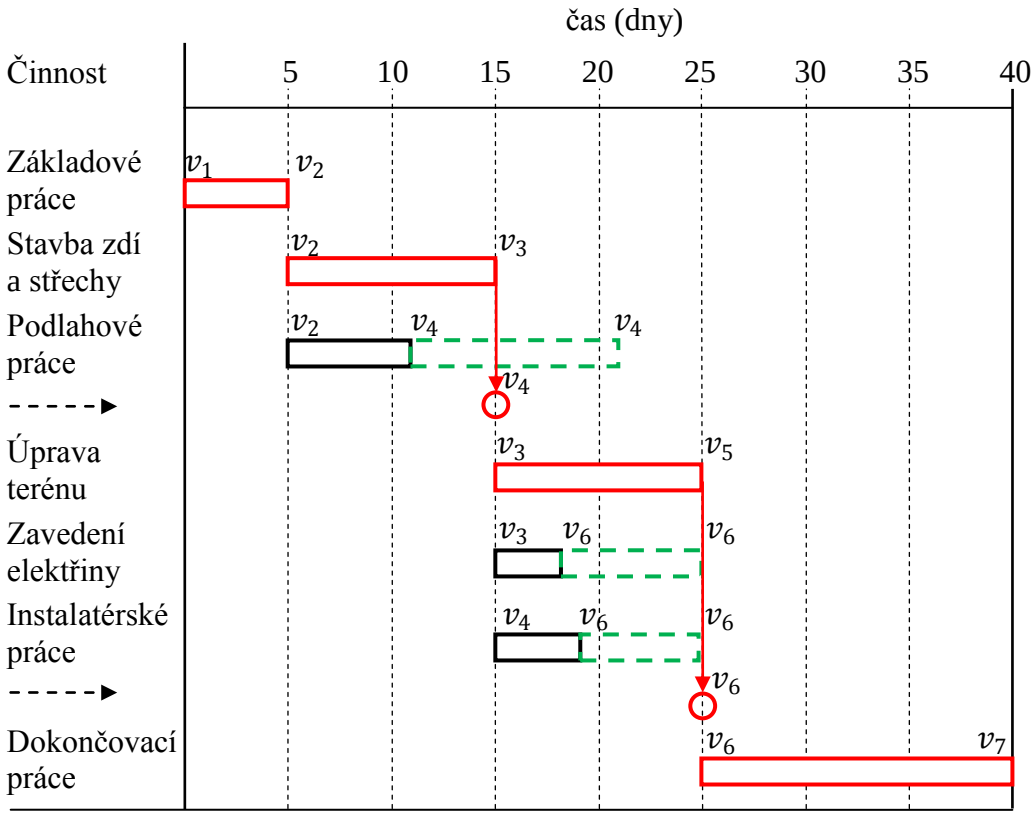
vytvořit síťový graf, pokud ovšem nemáme ještě nějaké doplňující informace. Pro sestrojení síťového grafu potřebujeme ještě znát kompletní informace o závislostech všech dílčích činností. Jednomu Ganttovu diagramu může odpovídat několik síťových grafů.

Existují některé úpravy základního Ganttova diagramu, které nám umožňují znázornit i závislosti v postupu realizace celého projektu, tak i zjištění vzniklých časových rezerv. Ganttův diagram zobrazující i závislosti mezi dílčími činnostmi se nazývá sledovací Ganttův diagram [2] [14].

4.2.2 Konstrukce Ganttova diagramu

Sestavení Ganttova diagramu si předvedeme na příkladu podle síťového grafu z obrázku 5. V prvním kroku seřadíme vzestupně činnosti podle koncových uzlů v_j. Pokud do jednoho uzlu vstupují 2 a více činností, pak je řádíme vzestupně podle počátečních uzlů v_i. Umístění činností probíhá v časovém měřítku. Pro každý počáteční vrchol činnosti, vyjma počáteční činnosti celého projektu, platí, že jej vkládáme pod vrchol bezprostředně předcházející činnosti, která má největší hodnotu nejdříve možného ukončení, tedy je ukončena nejpozději [2] [14].

Jako první vložíme činnosti $[v_1, v_2]$, která vystupuje z počátečního uzlu. Na tuto činnost dále navazují činnosti $[v_2, v_3]$ a $[v_2, v_4]$, takže jejich počáteční vrcholy umístíme pod vrchol v_2 činnosti $[v_1, v_2]$. Dalšími činnostmi, které lze nyní zakreslit jsou $[v_3, v_4]$, $[v_3, v_5]$, $[v_3, v_6]$ a $[v_4, v_6]$. Činnost $[v_3, v_4]$ je ovšem fiktivní, takže neprobíhá v čase. Lze ji zakreslit například vertikálou. Podle předchozího postupu vložíme do Ganttova diagramu činnosti $[v_3, v_5]$ a $[v_3, v_6]$. Aktivita $[v_4, v_6]$ má obě bezprostředně předchozí činnosti ukončené v odlišných časech, ale řídíme se pravidlem, že počáteční vrchol následuje za předchozí činností, která byla ukončena v čase jako poslední. Tuto činnost vložíme až pod koncový vrchol činnosti $[v_2, v_3]$. Následuje vložení fiktivní činnosti $[v_5, v_6]$, která se promítne v diagramu jako vertikála. Poslední činnost $[v_6, v_7]$, která může být vložena až po fiktivní činnosti $[v_5, v_6]$, která má ze všech bezprostředně předcházejících aktivit nejvyšší dobu nejdříve možného ukončení činnosti.



Obrázek 10 - Ganttův diagram stavby městského domu – Zdroj: vlastní

Jedním z prvků Ganttova diagramu je zobrazení kritické cesty, která je znázorněna červenou barvou. V diagramu lze například znázornit i časové rezervy činností. V našem příkladu jsou znázorněny celkové rezervy zelenou barvou. Z obrázku je vidět, že činnosti na kritické cestě mají celkovou časovou rezervu nulovou.

4.3 Zhodnocení síťového a Ganttova diagramu

Oba způsoby znázornění projektu jsou stále velmi využívány v projektovém řízení. Současné profesionální softwarové nástroje většinou obsahují obě varianty znázornění. Proto si nyní uvedeme základní rozdíly ve využití těchto zobrazení projektů.

Síťové grafy nabízejí grafický přehled o celkovém rozsahu projektu, jak na sebe jednotlivé činnosti navazují, nebo zdali mohou být vykonávány současně. V síťovém grafu lze zohlednit další atributy projektu, jako jsou například náklady, produktivita výroby, vytížení strojního zařízení a pracovníků. Pomocí síťových grafů lze rychle nalézt důležité činnosti, či naopak aktivity, které nemají takovou prioritu a přizpůsobit tomu pracovní síly. Jednou z nevýhod je nutná znalost problematiky každého pracovníka, který s diagramem musí pracovat.

Velkým přínosem Ganttových diagramů je přehled o časovém průběhu činností projektu. Díky této vlastnosti diagramy představují silný nástroj sloužící ke kontrole postupu prací. Ganttovy diagramy se vyznačují jednoduchostí a přehledností. V základní formě ovšem zanedbávají Ganttovy diagramy návaznosti činností mezi sebou. Z čehož vyplývá, že je nelze využít v oblastech, kde je kladen důraz na koordinaci. Diagramy nejsou rovněž vhodné tam, kde je nutné sledovat celkové náklady, či efektivitu práce a podobné ukazatele.

5 Metody síťové analýzy pro řízení projektů

Před uvedením do problematiky o rozdělení zdrojů pomocí síťové analýzy bychom se měli seznámit se dvěma stěžejními metodami pro řízení projektů, na kterých daná problematika staví. Obě metody používají podobné principy, avšak mají rozdílné pohledy na dobu trvání činností projektu.

5.1 Metoda kritické cesty (CPM)

Prvními kroky pro analýzu projektu je sestrojení síťového diagramu. Poté následuje výpočet lhůtových ukazatelů charakterizujících projekt. Pro tvorbu modelu projektu pomocí kritické cesty je zapotřebí znát tyto informace:

- Seznam všech aktivit, které jsou potřeba k dokončení projektu
- Časové odhady pro vykonání dílčích činností
- Jednotlivé vztahy mezi činnostmi

Tyto údaje slouží k výpočtu termínů pro zahájení a ukončení činností a celkový výpočet kritické cesty. Kritická cesta je posloupnost činností s nejdelší dobou trvání z počátečního uzlu diagramu do koncového uzlu. Zároveň určuje nejkratší možný čas trvání realizace celého projektu. Důsledkem jakékoliv prodlevy na kritické cestě znamená, že dojde k prodloužení celého projektu. Pro činnosti na kritické cestě tedy platí, že nemají žádnou časovou rezervu. Ostatní činnosti, které mají pružnou dobu pro jejich vykonávání, se nazývají nekritické činnosti. V praxi se stává, že metoda může mít i více řešení, tj. více kritických cest. Zbylé cesty z počátečního uzlu do koncového uzlu se nazývají nekritické [2].

Díky těmto ukazatelům, může projektant přidělit jednotlivým činnostem priority a zabránit, tak případnému prodloužení projektu, a tím i snížit celkové náklady. Zavedeme si značení pro lhůtové ukazatele plánu [1]:

$o[v_i, v_j]$ – doba trvání činnosti $[v_i, v_j]$,

t_i – nejdříve možný termín zahájení činnosti $[v_i, v_j]$,

t_i' – nejpozději přípustný termín zahájení činnosti $[v_i, v_j]$,

t_j – nejdříve možný termín ukončení činnosti $[v_i, v_j]$,

t_j' – nejpozději přípustný termín ukončení činnosti $[v_i, v_j]$.

5.1.1 Výpočet kritické cesty

Výpočet se skládá ze dvou etap. V první etapě postupujeme od počátečního uzlu ke koncovému a provádíme výpočet všech termínů nejdříve možných, ve druhé pak postupujeme od koncového uzlu k počátečnímu a propočítáváme termíny nejpozději přípustné [1].

1. krok: Hodnota celkového času projektu na jeho začátku je rovna $t_0 = 0$.

2. krok: Propočítáme časy nejdříve možných začátků dílčích činností podle vzorce:

$$t_j = \max_{h \in Y, p[h]=[v_i,v_j]} \left(t_i + o\left[v_i, v_j\right]\right).$$

3. krok: Poslední určená hodnota udává nejdříve možný čas t_n dokončení celého projektu, tedy hodnotu kritické cesty.

4. krok: Vypočítáme čas nejpozději možného ukončení celého projektu t_n', který je roven času nejdříve možného ukončení projektu $t_n = t_n'$.

5. krok: Určíme časy nejpozději nutného ukončení jednotlivých dílčích činností t_i'. Každému vrcholu $v_i \in V$ přiřadíme hodnotu $t_i' = \infty$, pro $i = 0,1,2, \dots, n-1$. Poté hledáme takovou orientovanou hranu $h \in Y, p[h] = \left[v_i, v_j\right]$, pro kterou platí:

$$t_j' - o\left[v_i, v_j\right] < t_i',$$

5a. krok: Pokud orientovaná hrana, která splňuje danou nerovnost existuje, pak můžeme určit její čas nejpozději nutného ukončení:

$$t_i' = t_j' - o\left[v_i, v_j\right],$$

5b. krok: Jestliže hrana splňující tuto podmínku neexistuje, pak pokračujeme krokem 6.

6. krok: Kritickou cestu $m[v_0, v_n]$ vytvoříme tak, že pro všechny činnosti určíme celkovou časovou rezervu podle vzorce:

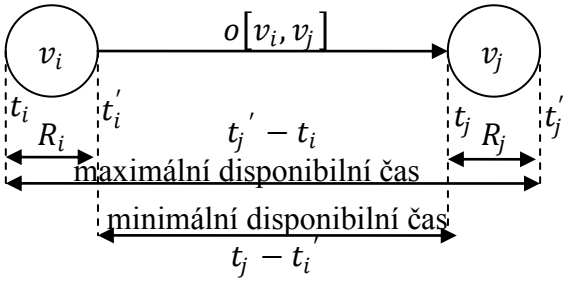
$$R_{ij}^c = t_j' - t_i - o\left[v_i, v_j\right].$$

Pro činnosti, které mají $R_{ij}^c = 0$ leží na kritické cestě.

Kritická cesta je libovolná dráha, jejíž součet ohodnocení hran t_n je maximální. Všechny činnosti na této dráze jsou **kritické činnosti**. Prodloužením libovolné kritické činnosti o e dojde k prodloužení celkové doby pro dokončení projektu $t_n = t_n + e$ [1].

5.1.2 Časové rezervy

Pokud činnost neleží na kritické cesty, pak to znamená, že má k dispozici více času, než je potřeba pro její vykonání. Z obrázku 7 je patrné, že maximální disponibilní čas činnosti je roven D_{ij} – maximální disponibilní čas, $D_{ij} = t_j{}' - t_i$. Tento rozdíl hodnot nabývá vždy nezáporných hodnot, jelikož $t_j{}' \geq t_i$.



Obrázek 11 - Lhůtové ukazatele CPM [2]

Jestliže daná činnost má více času pro zpracování než je její doba trvání, pak říkáme, že daná činnost má časovou rezervu, kterou nazýváme celková časová rezerva a její hodnota je:

$$R_{ij}^c = D_{ij} - o[v_i, v_j]. \text{ [2]}$$

Jak je již uvedeno v algoritmu pro vytvoření kritické cesty, všechny činnosti, které mají $R_{ij}^c = 0$, jsou kritické a leží na kritické cestě.

Pro události (uzly) patřící kritické cestě platí, že hodnota $R_i = t_i{}' - t_i$, kde R_i nazýváme interferenční rezervou *i*–tého uzlu, respektive pro *j*-tý uzel je tato interferenční rezerva dána vztahem $R_j = t_j{}' - t_j$. Všechny uzly ležící při kritické cestě mají interferenční rezervu rovnu nule. Ovšem, pokud existuje činnost $[v_i, v_j]$, pro jejíž uzly platí, že $R_i = 0$ a $R_j = 0$, nemusí to znamenat, že činnost je kritická.

Kromě interferenčních rezerv a celkových rezerv se v síťové analýze velmi často používají další dvě rezervy. Jednou z těchto rezerv je volná časová rezerva [2]:

$$R_{ij}^v = t_j - t_i - o[v_i, v_j],$$

která má tu vlastnost, že čerpání této rezervy nemá vliv na činnosti síťového

diagramu, které následují dané činnosti $[v_i, v_j]$.

Podle obrázku 7 lze určit minimální disponibilní čas činnosti, který je dán vztahem $d_{ij} = t_j - t_i{}'$. Pokud je hodnota $d_{ij} > o[v_i, v_j]$, pak má činnost nezávislou časovou rezervu [2]:

$$R_{ij}^n = d_{ij} - o[v_i, v_j].$$

Činnost, která má hodnotu $R_{ij}^n > 0$, můžeme o tuto dobu prodloužit, aniž by to mělo vliv na trvání celého projektu nebo kterékoliv dílčí činnosti síťového diagramu. Nezávislá rezerva může nabývat i záporných hodnot. V tomto případě záporné hodnoty ignorujeme a pokládáme $R_{ij}^n = 0$.

Pro všechny rezervy činností platí, že $R_{ij}^c \geq R_{ij}^v \geq R_{ij}^n$. V případě, kdy leží oba uzly činnosti na kritické cestě, avšak činnost $[v_i, v_j]$ není kritická, pak jsou všechny tři časové rezervy shodné: $R_{ij}^c = R_{ij}^v = R_{ij}^n$. V následující tabulce je uveden přehled časových rezerv a vztahů mezi nimi.

	$\boldsymbol{R_{ij}^c}$	$\boldsymbol{R_{ij}^v}$	$\boldsymbol{R_{ij}^n}$
$\boldsymbol{R_{ij}^c}$	$t_j{}' - t_i - o[v_i, v_j]$	$R_{ij}^v + R_j$	$R_{ij}^n + R_i + R_j$
$\boldsymbol{R_{ij}^v}$	$R_{ij}^c - R_j$	$t_j - t_i - o[v_i, v_j]$	$R_{ij}^n + R_i$
$\boldsymbol{R_{ij}^n}$	$R_{ij}^c - R_i - R_j$	$R_{ij}^v - R_i$	$t_j - t_i{}' - o[v_i, v_j]$

Tabulka 2 - Vztahy mezi časovými rezervami [2]

5.2 Metoda PERT

Další metodou síťové analýzy je systém PERT. U metody kritické cesty dochází zpravidla k odhadování doby trvání jednotlivých činností. S tím je spojen určitý stupeň nejistoty. Mnohdy se nelze opřít o zkušenosti z minulosti a časové odhady aktivit s sebou přináší určitou míru rizika. S větší rozsáhlostí projektu dochází zároveň ke zvýšení stupně nejistoty a rizik, které jsou spojeny s jeho realizací. Některé projekty vyžadují, aby byl brán v potaz určitý prvek nahodilosti, který dokáže kvantifikovat stupeň nejistoty.

Z tohoto důvodu byl vytvořen systém PERT, který pohlíží na činnost jako na náhodnou proměnnou s určitým rozdělením pravděpodobnosti. Tímto způsobem

pak lze měřit míru nejistoty. Systém prohlubuje pojetí kritické cesty, jelikož zde přibývají další hlediska pro identifikaci potencionálně kritických činností.

5.2.1 Časové odhady

Velmi často nastává případ, kdy pracovníci nedokážou dostatečně odhadnout dobu trvání vykonávání činnosti. Takovýto odhad je značně nejistý. Z tohoto důvodu metoda PERT pracuje se třemi časy, kterými má pracovník možnost určit časový odhad trvání činnosti. Dvěma krajními hodnotami určí ohraničení intervalu, ve kterém by se mělo časové ohodnocení činnosti nacházet. Třetí hodnotou je pak odhad, kdy lze s největší pravděpodobností dokončení činnosti očekávat [2].

Optimistický odhad trvání činnosti je nejkratší možná doba trvání činnosti. Pro jeho dodržení je zapotřebí mimořádná souhra událostí a velká míra štěstí. Pravděpodobnost splnění tohoto termínu by měla zhruba být 1:100. Tento časový odhad se značí a.

Odhad nejpravděpodobnější doby trvání činnosti, který značíme m. Při opakování vykonávání téže činnosti za totožných podmínek, by s největší pravděpodobností nastal nejčastěji právě tento časový odhad. Z tohoto poznatku vyplývá, že se jedná o modus příslušného rozdělení pravděpodobnosti.

Pesimistický odhad b trvání činnosti je doba, která určuje nejpozději pravděpodobné ukončení činnosti. Zohledňují se zde mnohá rizika zvyšující délku trvání činnosti. Při opakovaném provádění činnosti by mělo k tomuto odhadu teoreticky dojít pouze v jednom případu ze sta.

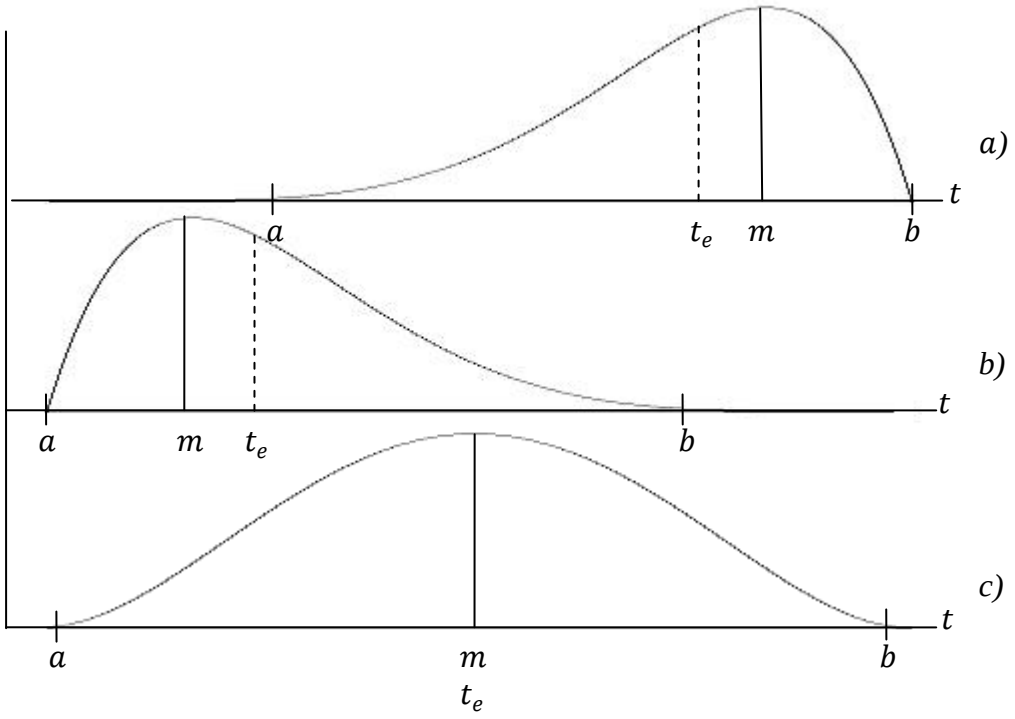
5.2.2 Odvození střední hodnoty doby trvání činnosti a směrodatné odchylky

V praxi se dále již nepracuje se všemi třemi odhady, ale transformujeme je v jediný odhad střední hodnoty doby trvání činnosti, který značíme t_e. S touto hodnotou dále pracujeme pro určení lhůtových ukazatelů projektu.

Pomocí těchto odhadů můžeme zkonstruovat hypotetickou křivku funkce hustoty pravděpodobnosti. Průběhy této funkce lze vyjádřit ve třech základních variantách. Systém PERT předpokládá, že náhodná veličina, kterou je doba trvání činnosti, má rozdělení beta.

V případě, že $t_e = m$, pak je křivka hustoty pravděpodobnosti symetrická, jak je patrné z následujícího obrázku. Pokud jsou tyto hodnoty rozdílné, pak má pe-

simistický odhad pracovníky dán vysokou hodnotou spjatou s případnými riziky.



Obrázek 12 - Hustota pravděpodobnosti Beta rozdělení [2]

Ačkoliv jsou modus m a rozpětí $(b - a)$ důležitými charakteristikami, nelze je s nimi dále pro výpočty lhůtových ukazatelů pracovat. Tyto údaje představují výchozí hodnoty pro výpočet střední hodnoty t_e, směrodatné odchylky σ_{t_e}, či rozptylu $\sigma_{t_e}^2$.

Funkce hustoty pravděpodobnosti je dána vztahem:

$$f(x) = \frac{1}{B(p,q)} x^{p-1}(1-x)^{q-1} \text{ pro } 0 \leq x \leq 1,$$

kde $B(p,q) = \frac{\Gamma(p)\Gamma(q)}{\Gamma(p+q)}$ vycházející ze vztahu funkce beta k funkci gama. [2]

Distribuční funkce je dána vztahem:

$$F(x) = \frac{1}{B(p,q)} \int_0^1 x^{p-1}(1-x)^{q-1}\, dx. \text{ [2]}$$

Pro určení t_e byla hodnota směrodatné odchylky pro B-rozdělení aproximována jednou šestinou rozpětí s využitím pravidla tří sigma, podle kterého lze u náhodné proměnné s normálním rozdělením s pravděpodobností 0,997 očekávat, že

bude patřit do intervalu $\langle \mu - 3\sigma, \mu + 3\sigma \rangle$:

$$\sigma_{t_e} = \frac{b-a}{6}, [1]$$

a výsledný rozptyl je dán vztahem :

$$\sigma_{t_e}^2 = \frac{(b-a)^2}{36}, [1]$$

Výpočtem počátečního momentu z rovnice rozdělení hustoty pravděpodobnosti B a po dalších dílčích úpravách získáme vzorec pro očekávaný čas doby trvání činnosti:

$$t_e[h] = \frac{a + 4m + b}{6}. [1]$$

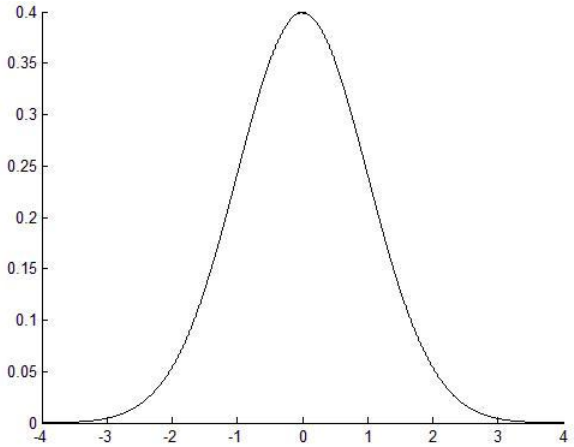
Tímto způsobem jsme získali časový odhad činnosti, se kterým dále můžeme pracovat analogicky jako s dobou trvání u metody kritické cesty.

5.2.3 Výpočet hodnot, $T_E^{v_i}$, $T_L^{v_i}$,$\sigma_{T_E^{v_i}}$, $\sigma_{T_L^{v_i}}$ a R^{v_i}

Výpočet časových hodnot $T_E^{v_i}$ a $T_L^{v_i}$ je shodný jako u metody kritické cesty. Rozdíl je v tom, že časy v uzlech jsou náhodné proměnné blížící se k normálnímu rozdělení $N(\mu, \sigma^2)$, kde μ je střední hodnota rozdělení a σ^2 rozptyl. Tento předpoklad se opírá o zákon velkých čísel, že při velkém počtu nezávislých pokusů lze s velkou jistotou očekávat, že relativní četnost se bude blížit určitému rozdělení.

Hustota pravděpodobnosti normálního rozdělení je dána vzorcem:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}. [1]$$



Obrázek 13 - Hustota pravděpodobnosti normálního rozdělení Zdroj:vlastní

$T_E^{v_i}$ a $T_L^{v_i}$ jsou středními hodnotami normálního rozdělení. $T_E^{v_i}$ představuje délku maximální dráhy z počátečního uzlu v_0 do uzlu v_i. Daný vztah je pak:

$$T_E^{v_i} = \sum_{h \in \tilde{m}[v_0, v_i]} t_e[h] \,.\, [1]$$

Směrodatná odchylka času $T_E^{v_i}$ je pak analogicky vyjádřena:

$$\sigma_{t_e^{v_i}} = \left(\sum_{h \in \tilde{m}[v_0, v_i]} \sigma_{t_e}^2[h] \right)^{\frac{1}{2}} .\, [1]$$

$T_L^{v_i}$ pak představuje délku dráhy z uzlu v_i do koncového uzlu v_n:

$$T_L^{v_i} = T_L^{v_n} - \sum_{h \in \tilde{m}[v_i, v_n]} t_e[h] \,.\, [1]$$

Pro směrodatnou odchylku času $T_L^{v_i}$ platí vztah:

$$\sigma_{T_L^{v_i}} = \left(\sum_{h \in \tilde{m}[v_i, v_n]} \sigma_{t_e}^2[h] \right)^{\frac{1}{2}} .\, [1]$$

Časy T_E a T_L jsou střední hodnoty dvou na sobě nezávislých normálních rozdělení, jejichž křivky hustoty pravděpodobnosti jsou závislé na $\sigma_{T_E}^2$ a $\sigma_{T_L}^2$. Čím vyšší je hodnota směrodatné odchylky, tím má křivka hustoty pravděpodobnosti plošší tvar a opačně (viz. obrázek 14).

Hodnota interferenční rezervy v uzlech se může zvětšovat nebo zmenšovat, respektive může nabývat i záporných hodnot. Tento případ nastane tehdy, jestliže se obě křivky protínají (viz. obrázek 14). Pouze tehdy hodnota $T_E^{v_i}$ může být větší než $T_L^{v_i}$. Potom platí vztah:

$$T_L^{v_i} - T_E^{v_i} = -R. \, [2]$$

V takovém případě se mohou dosavadní nekritické činnosti změnit v kritické a mohou tak ohrozit včasné splnění celého projektu. Z tohoto důvodu je vhodné testovat i nekritické uzly a určit s jakou pravděpodobností se mohou změnit v uzel kritický, tj. $P\{R \leq 0\}$.

Určíme podíl rezervy se záporným znaménkem a druhé odmocniny součtu rozptylů obou rozdělení, získáme *u*, což je náhodná veličina *N*(0, 1):

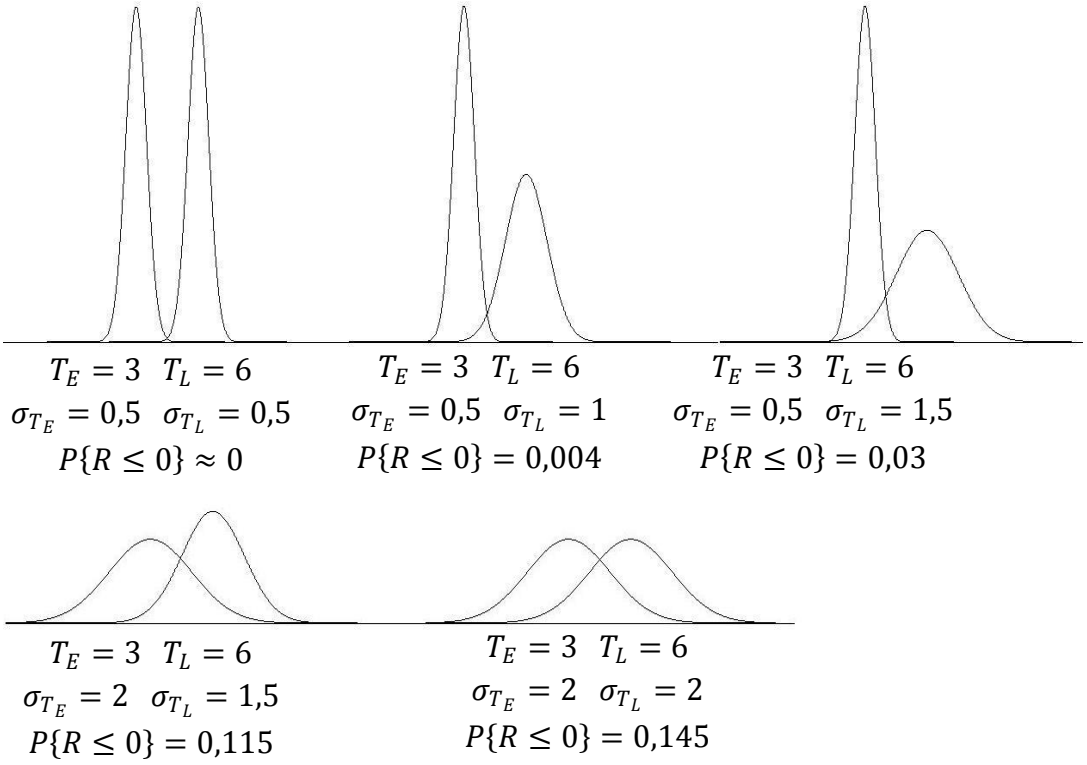
$$u = \frac{T_E - T_L}{\sqrt{\sigma_{T_E}^2 + \sigma_{T_L}^2}} = \frac{-R}{\sigma_{T_R}}, \text{jelikož platí}$$

$$\sigma_{(T_E - T_L)}^2 = \sigma_{T_E}^2 + \sigma_{T_L}^2 = \sigma_{T_R}^2 . [1]$$

Hodnoty pravděpodobnosti zjistíme v tabulce distribuční funkce normálního rozdělení *N*(0, 1):

$$P\{R \leq 0\} = \Phi(\mathrm{u}) = \Phi\left(\frac{-R}{\sigma_{T_R}}\right). [1]$$

Na následujícím obrázku je uveden příklad vzniku záporné rezervy u uzlu, jehož interferenční rezerva má hodnotu ve výši tří časových jednotek. Na obrázku je pět variant, u kterých zůstává hodnota interferenční rezervy stejná. Mění se pouze hodnoty směrodatných odchylek σ_{T_E}, σ_{T_L}.



Obrázek 14 - Pravděpodobnost vzniku záporné rezervy – Zdroj: [2]

Hodnota *u* je závislá na velikosti časové rezervy, tak i na velikosti směrodatných odchylek obou rozdělení. S rostoucími směrodatnými odchylkami roste rychle

i pravděpodobnost vzniku záporné mezery, i když interferenční rezerva v uzlu zůstane nezměněna. Pro všechny uzly na kritické cestě platí, že lze s 50% pravděpodobností očekávat, že se u nich vytvoří časová rezerva, která příslušnou činnost změní v nekritickou.

5.2.4 Pravděpodobnost dodržení plánovaného termínu

Velký význam hraje odhad pravděpodobnosti dodržení plánovaných termínů, a to nejen ukončení celého projektu, ale i jeho významných etap.

Je–li pro určitý uzel dán termín $T_S^{v_i}$, potom platí pro odhad pravděpodobnosti vztah $P\{T_E^{v_i} = T_S^{v_i}\}$. Nyní položíme $T_S^{v_i} = T_L^{v_i}$ a hodnotu $T_S^{v_i}$ dosadíme do vzorce pro výpočet u:

$$u' = \frac{T_E^{v_i} - T_L^{v_i}}{\sqrt{\sigma^2_{T_E^{v_i}} - \sigma^2_{T_L^{v_i}}}} = \frac{T_E^{v_i} - T_S^{v_i}}{\sigma_{T_E^{v_i}}}, \text{neboť } \sigma_{T_S^{v_i}} = 0. [1]$$

Z rozdílu $T_E^{v_i} - T_S^{v_i}$ plyne, že hodnota u' vede k odhadu pravděpodobnosti překročení plánovaného termínu $T_S^{v_i}$.

Jestliže má náhodná proměnná představující čas v uzlu normální rozdělení $N\left(T_E^{v_i}, \sigma^2_{T_E^{v_i}}\right)$, pak pro rozdíl $T_S^{v_i} - T_E^{v_i}$ odhadujeme pravděpodobnost dodržení termínu $T_S^{v_i}$:

$$P\{T_E^{v_i} \leq T_S^{v_i}\} = \Phi(u) = \Phi\left(\frac{T_S^{v_i} - T_E^{v_i}}{\sigma_{T_E^{v_i}}}\right). [1]$$

- Jestliže $T_S^{v_i} > T_E^{v_i}$ je pravděpodobnost dodržení termínu $p > 0{,}5$.
- Jestliže $T_S^{v_i} = T_E^{v_i}$ je pravděpodobnost dodržení termínu $p = 0{,}5$.
- Jestliže $T_S^{v_i} < T_E^{v_i}$ je pravděpodobnost dodržení termínu $p < 0{,}5$.

Tímto odhadem pravděpodobnosti určujeme dodržení nejen plánovaného ukončení celého projektu, ale i jeho významných událostí v některých uzlech síťového diagramu. V praxi se uvádí, že pokud hodnota pravděpodobnosti uzlu klesne pod 0,25, pak by mělo dojít na všech činnostech, které leží na cestě z počátečního do daného uzlu, k určitým opatřením vedoucích ke zlepšení prací této cesty [2].

6 Problém rozmístění pracovních sil

Cílem je nalézt taková řešení, která minimalizují dobu trvání celého projektu a zároveň respektují kapacitní omezení zdrojů. Jedná se o náročný kombinatorický problém diskrétní optimalizace, kdy je nutné prozkoumat velké množství možností. Tento problém si ukážeme na následujícím příkladě stavby městského domu z kapitoly o konstrukci síťového grafu, který rozšíříme o pracovníky, kteří jsou nutní pro vykonání jednotlivých činností.

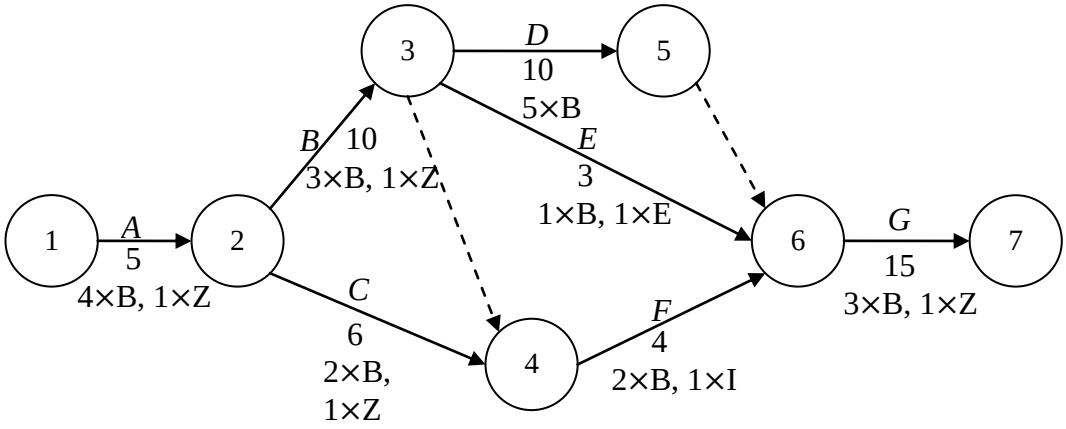
Činnost	Uzly	Ohodnocení	Pracovníci
A	1-2	5	1 zedník, 4 brigádníci
B	2-3	10	1 zedník, 3 brigádníci
C	2-4	6	1 zedník, 2 brigádníci
D	3-5	10	5 brigádníků
E	3-6	3	1 elektrikář, 1 brigádník
F	4-6	4	1 instalatér, 2 brigádníci
G	6-7	15	1 zedník, 3 brigádníci

Tabulka 3 - Zdroje stavby městského domu – Zdroj: vlastní

Pracovní síly jednotlivých činností jsou uvedeny v předcházející tabulce. V síťovém grafu jsou pak tyto zdroje označeny zkratkami (brigádník – B, zedník – Z, elektrikář – E, instalatér – I). Nad každou činností je uvedena její zkratka. Pod jednotlivými pracemi je pak uvedeno časové ohodnocení a pod ním pracovníci, kteří jsou potřební pro jejich vykonání. K dispozici máme 7 brigádníků, 2 zedníky, 1 elektrikáře a 1 instalatéra.

Nyní potřebujeme prozkoumat, kdy lze dané práce vykonat, tak aby byl dostatek pracovníků. Projekt lze začít pouze činností *A*. V dalším kroku je možné vykonat činnosti *B* a *C* současně, jelikož máme pro ně dostatek pracovních sil nebo každou činnost zvlášť. Tímto jsme získali 3 možná řešení postupu. Pokud bychom zvolili, že nyní vykonáme činnost *B*, pak v další fázi získáme možnost zpracovat samostatně aktivity *C, D, E*, nebo současně činnosti *C, D*, nebo činnosti *C, E*, nebo činnosti *D, E*. Činnosti *C, D, E* nelze vykonávat souběžně, jelikož je pro všechny zapotřebí mít k dispozici 8 brigádníků. Po tomto kroku nám přibylo dalších 8 variant, které následně musíme prozkoumat. Nyní je nutné se vrátit do kroku po ukončení činnosti *A* a prozkoumat zbylé možnosti (činnost *C* nebo souběžně činnosti *B* a *C*).

Takovýmto způsobem bychom pokračovali dále. Obecně se udává u podobných úloh, že v každém kroku se může hledání optimálního řešení větvit na *n* dalších podproblémů.



Obrázek 15 - Síťový graf stavby městského domu se zdroji – Zdroj: vlastní

V případě řešení pomocí síťové analýzy lze určit, na kolik podproblémů se může daná úloha v každém kroku větvit. Vytvoříme si proměnnou $n = st(v_i^+)$ určující počet bezprostředně následujících činností pro uzel v_i. V krajním případě, kdy je v libovolném čase projektu dostatek zdrojů, se může v uzlu v_i podproblém rozšířit o dalších:

$$\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1$$

podproblémů. Z tohoto jevu lze vyčíst, že pokud bude při hledání optima přibývat mnoho podproblémů již na začátku úlohy, pak jejich počet bude stoupat výrazně rychleji a nabude vyššího počtu.

Takováto úloha je označována jako NP-těžká a její výpočetní složitost je exponenciální. Proto je nutné zkoumat jen některá řešení.

6.1 Algoritmy pro řešení NP-těžkých problémů

Velké množství úloh je NP-úplných. Nalézt pro ně efektivní polynomiální algoritmus lze pouze za předpokladu P = NP. Ovšem pokud máme úlohu P ≠ NP, pak polynomiální algoritmus neexistuje. Přesto máme možnost využít následujících dvou základních skupin algoritmů [4].

Deterministické algoritmy po určitém počtu kroků dojdou k výsledku. Jejich

nespornou výhodou je to, že vždy naleznou optimální řešení. Tyto algoritmy jsou velmi dobře popsány a z tohoto důvodu se velmi často aplikují. Největším záporem těchto metod je neefektivnost, kdy pro rozsáhlejší úlohy nelze v dostatečně rychlém čase dospět k řešení. Důvodem je rostoucí výpočetní náročnost, která roste exponenciálně. V této práci se budeme zabývat z deterministických metod pouze metodou větví a mezí.

Druhou skupinu tvoří **heuristické algoritmy**. Rozvoj těchto algoritmů přišel s rozmachem počítačových technologií. Jejich výhodou je nalezení uspokojivého řešení v přijatelném čase. Nedostatkem těchto metod je, že nemusí vést k nalezení optimálního řešení. Některé z heuristických algoritmů využívají principu gradientních metod, kdy se postupuje ve směru největšího poklesu účelové funkce. Nastává zde problém, stejně jako u gradientních metod, že při hledání řešení mohou uváznout v lokálním extrému. Důvodem je, že gradientní metody konvergují pouze v blízkosti počáteční bodu. Tento problém se snaží tyto algoritmy řešit jistou mírou stochastičnosti. Takovéto algoritmy, využívající heuristické strategie s jistou dávkou nahodilosti při hledání řešení, jsou v mnohých literaturách označovaný jako **metaheuristické.** Hlavním principem metaheuristických algoritmů je nalézt, co nejrychleji lokální optimum a co nejrychleji prohledat prostor řešení.

6.2 Metoda větví a mezí (Branch and Bound)

Velké skupina plánovacích úloh patřících do skupiny kombinatorických optimalizačních problémů má několik společných rysů. Optimalizují řešení, staví na jednoduchých principech, ale zpravidla je u nich nutné řešit velké množství proveditelných řešení. Tyto úlohy jsou známé jako NP-těžké.

Metoda větví a mezí je jedním z nejrozšířenějších nástrojů pro hledání řešení optimalizačních kombinatorických problémů. Řešení NP-těžkých úloh diskrétní optimalizace vyžaduje velmi účinné algoritmy a jedním z nich je paradigma metody větví a mezí. Metoda prohledává velkou část prostoru řešení, abychom získali pro daný problém nejlepší řešení. Nicméně explicitní výpis všech možností je nereálný z důvodu exponenciálního růstu počtu potencionálních řešení. Použitím hraniční funkce spolu v kombinaci s aktuálně nejlepším řešením umožňuje algoritmu prohledat část prostoru řešení implicitně.

V každém kroku v průběhu hledání optima je stav řešení problému popsán

neprozkoumanou podmnožinou prostoru řešení a doposud nejlepší nalezenou hodnotou. Na začátku celé úlohy máme pouze jedno jediné řešení, které je reprezentováno jako kořen stromu. Tento strom nese specifický název **prastrom řešení**. Neprozkoumané podmnožiny prostoru řešení jsou znázorněny jako listy prastromu. V jednotlivých krocích základní metody větví a mezí je zpracováván pouze jediný list, který je vybrán podle určené strategie. Všechny iterace se skládají ze třech hlavních kroků:

- výpočet hranice pomocí účelové funkce,
- výběr uzlu pro zpracování podle námi vybrané strategie, čili pořadí, v jakém budou dosud neprozkoumané podproblémy zpracovány,
- větvení konkrétního podproblému na 1 až *n* větví.

Sled těchto iterací se může lišit na zvolené strategii pro výběr uzlu pro zpracování. Jestliže je pro výběr dalšího podproblému použita hodnota hraniční funkce, pak dalším krokem iteraci je větvení, tj. rozdělení prostoru řešení do dvou, či více dalších podmnožin, které mají být zkoumány v dalších iteracích. Pro každou z podmnožin je nutné zjistit, zda se skládá z jediného řešení, které se porovnává zatím s aktuálně nejlepším. V opačném případě se pomocí hraniční funkce určí hodnota pro danou podmnožinu a je porovnána s nejlepším dosavadním řešením. Pokud by bylo zřejmé, že daná podmnožina řešení neobsahuje optimum, pak by došlo k jejímu zamítnutí a dále by nebyla prozkoumávána. Tato strategie je nazývána jako „horlivá". Alternativní strategií je začít výpočtem hraniční funkce vybraného uzlu a následně jeho případné větvení. Nově vytvořené uzly jsou pak ukládány spolu s hranicí zpracovávaného uzlu. Této strategii byl přiřazen přívlastek „líná" a využívá se zejména tam, když má být další uzel vybrán tak, abychom dosáhli, co největší hloubky v prastromu řešení. Tento postup má za cíl, co nejrychleji nalézt alespoň jedno optimální řešení. Algoritmus končí tehdy, když ve zbylém neprozkoumaném podprostoru řešení už nemůže být nalezeno lepší řešení, než je aktuálně nejlepší [15].

6.2.1 Prastrom řešení

Řešení problému metodou větví a mezí je typicky popsáno prohledáváním prastromu řešení, ve kterém kořen představuje původní neřešený problém. Každý další uzel představuje podproblém původní úlohy. Listy prastromu odpovídají přípustnému řešení a pro všechny NP-těžké úlohy platí, že počet listů dosahuje expo-

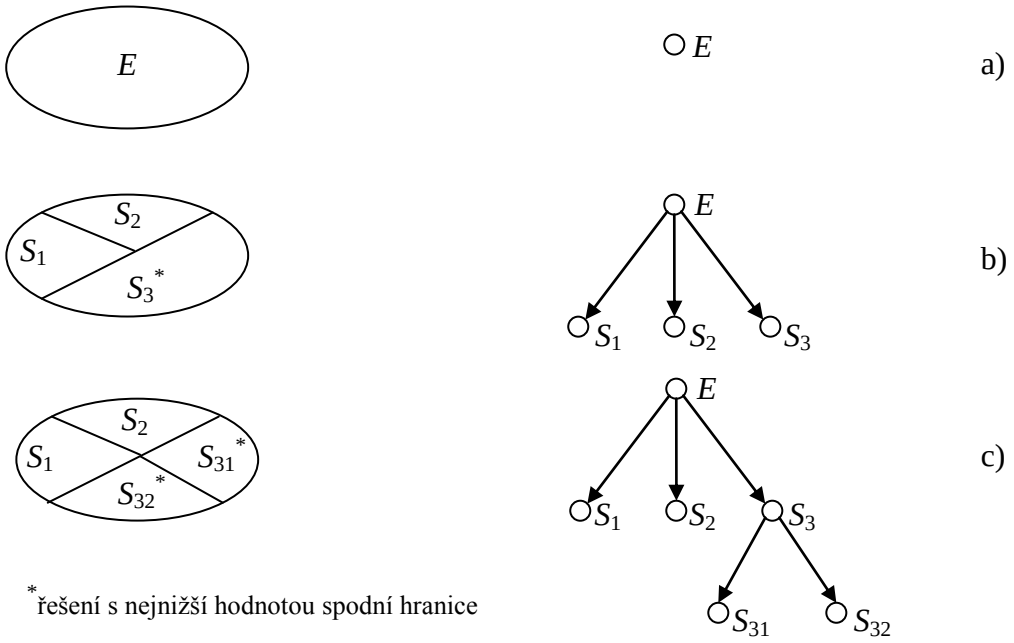
nenciálního růstu. Pomocí hraniční funkce g můžeme každému uzlu přiřadit reálné číslo, které nazýváme hranicí uzlu. Pro listy tato hodnota znamená odpovídající řešení, zatímco pro vnitřní uzly je hodnota rovna spodní hranici kteréhokoliv řešení v podprostoru příslušného uzlu. Pro funkci g a účelovou funkci f platí tři následující podmínky [15]:

- $g(P_i) \leq f(P_i)$ pro všechny uzly P_i prastromu řešení,
- $g(P_i) = f(P_i)$ pro všechny listy prastromu řešení,
- $g(P_i) \geq g(P_j)$ kde uzel P_j je otcem uzlu P_i.

Ve všech listech prastromu jednotlivé stavy funkce g jsou shodné s účelovou funkcí f a které nám poskytují bližší informace pro řešení podproblému. Prastrom řešení obsahuje při inicializaci pouze kořen a konstrukce prastromu je vytvářena dynamicky při prohledávání prostoru řešení. V každé iteraci metody větví a mezí je prozkoumán, dle zvolené strategie, vybraný list prastromu odpovídající neprozkoumané podmnožině prostoru řešení. Pokud je zvolena horlivá strategie, pak pro daný uzel jsou zkonstruovány dva a více potomků. Podmnožina prostoru řešení je tak rozdělena na více menších podmnožin. Pro nově vytvořené uzly se vypočítá hodnota hranice. V případě, že hodnota hranice uzlu je přípustná, pak je porovnána s dosavadním nejlepším řešením problému. Jestliže hodnota není lepší než prozatím nejlepší nalezené řešení, pak je tento list dočasně zamítnut, dokud v dané podmnožině prostoru řešení určené daným uzlem nemůže obsahovat lepší hodnotu. Když nastane situace, kdy žádný z potomků uzlu neobsahuje přípustné řešení, pak je celá podmnožina řešení příslušného uzlu zamítnuta. Ve strategii ‚líné" verze výběru uzlu je pořadí výpočtu hranice a větvení opačné.

Na následujícím obrázku je příklad vytváření prastromu řešení. V kroku a) množina E představuje původní problém a jemu odpovídá kořen prastromu. V následujícím kroku jsou možná 3 řešení dané úlohy S_1, S_2, S_3, která jsou zároveň přípustná. Vybereme řešení (uzel) s nejnižší hodnotou spodní hranice. Nejnižší hodnotu má pouze řešení S_3. Pokud by bylo řešení více, pak můžeme uplatnit další strategie, například nejvíce dokončených činností předcházejících podproblému apod. V kroku c) vytvoříme opět nová řešení, která jsou přípustná. Vznikla nám 2 nová řešení. Opět aplikujeme výběr uzlu dle zvolené strategie (v dalším kroku to může být

buď uzel S_{31} nebo S_{32}, jelikož oba mají stejnou hodnotu spodní hranice) a analogicky pokračujeme dokud nezískáme optimální řešení úlohy.



Obrázek 16 - Konstrukce prastromu – Zdroj: vlastní

6.2.2 Algoritmus metody větví a mezí

Jako příklad algoritmu si uvedeme metodu větví a mezí s „horlivou" strategií, kdy nejdříve dochází k větvení a posléze k výpočtu dolní hranice účelové funkce [15].

1. Inicializace: $f(x_{nejlepší}) = \infty$; spodní hranice $b_0 = g(P_0)$; množina neprozkoumaných řešení $E = \{(P_0, b_0)\}$.
2. Pokud je množina $E = \{\emptyset\}$, pak algoritmus končí.
3. Vybereme uzel P pro zpracování dle námi určeného kritéria. A odebereme tento uzel z množiny neprozkoumaných řešení $E = E \setminus \{P\}$.
4. Pomocí větvení vygenerujeme uzly $P_1, \dots, P_k$ reprezentující podproblémy.
5. Pro všechny uzly $P_i, kde\ i = 1, \dots, k$ vypočítáme hodnotu spodní hranice $b_i = g(P_i)$.
6. Jestliže $b_i = f(x)$, kde x je přípustné řešení a současně $f(x) < f(x_{nejlepší})$, pak $x_{nejlepší} = x$.

7. Pokud $b_i \geq f(x_{nejlepší})$, pak je toto řešení dočasně zamítnuto, jinak $E = E \cup \{(P_i, b_i)\}$.

8. Návrat ke kroku 2.

6.2.3 Matematický popis metody větví a mezí

Nechť $E = \{S_1, S_2, S_3 \ldots, S_n\}$ je množina diskrétní optimalizace a *f* je účelová funkce definována na této množině. Při minimalizaci účelové funkce *f* musíme najít takovou podmnožinu $E_m \in E$, na které tato funkce obsahuje minimum. Alternativním způsobem lze účelovou funkci maximalizovat.

S vlastností P_A můžeme množinu *E* rozdělit na dvě podmnožiny A, $\bar{A}$, pro které platí $A \cap \bar{A} = \{\emptyset\}$ a $A \cup \bar{A} = E$.

Najdeme spodní mez b_0 hodnot funkce *f* na množině *E*. Dále předpokládejme, že dokážeme určit i spodní meze $b_1 \geq b_0, b_1' \geq b_0$, funkce *f* na podmnožinách A, $\bar{A}$.

Vlastnosti P_B, P_C dovolují rozložit množinu *E* na dvě části. Pak vlastnostem $P_A \wedge P_B, \overline{P_A} \wedge P_B, P_A \wedge \overline{P_B}, \overline{P_A} \wedge \overline{P_B}$ odpovídají podmnožiny $A \cap B, \bar{A} \cap B, A \cap \bar{B}, \bar{A} \cap \bar{B}$. Každá z těchto podmnožin představuje uzel grafu. Tyto jednotlivé uzly umožňují vytvořit prastrom řešení.

Prastrom řešení má tu vlastnost, že pokud sjednotíme všechny uzly, které jsou v dané fázi listy prastromu, získáme původní množinu řešení E . Je-li výsledkem uzel, který je listem, potom funkce f nabývá na této množině minimální hodnotu [1].

6.3 Heuristické metody

6.3.1 Metoda lokálního prohledávání (Local Search)

Nejjednodušší z heuristických metod je metoda lokálního prohledávání. Její nevýhodou je značná neefektivnost. Základním principem je prohledávání nejbližšího sousedství. Metoda určí hodnoty z okolního sousedství a porovná je s aktuálně nejlepším řešením. Nastává zde problém, že metoda může snadno uváznout v lokálním optimu a její výsledky mohou být neuspokojivé. Popíšeme si zde dvě varianty postupu prostorem řešení.

Postup lokálního prohledávání, kde naším cílem je hledání minimální účelové funkce:

1. Zvolíme náhodně počáteční řešení $x_0 \in X$, které je přípustné. Položí-

me $x_{nejlep\ší} = x_0$, $k = 0$.

2. Pro všechna sousední řešení $x_{k+1} \in N(x_k)$, kde N je nejbližší okolí a určíme nejnižší hodnotu účelové funkce $f(x_{k+1})$ z okolí N.
3. Pokud nevyhovuje žádný prvek, pak algoritmus končí.
4. Jestliže $f(x_{k+1}) < f(x_{nejlep\ší})$, potom $x_{nejlep\ší} = x_{k+1}$, zvýšíme iteraci $k = k + 1$ a pokračujeme krokem 2.
5. Pokud $f(x_{k+1}) \not< f(x_{nejlep\ší})$, pak je algoritmus ukončen.

Princip druhé varianty spočívá v tom, že se prohledávají sousední řešení postupně a pokud je hodnota účelové funkce lepší, pak se přesuneme do tohoto řešení a z něj se opět prozkoumává jeho nejbližší okolí. Jedním ze způsobů, jak kompenzovat uváznutí v lokálním extrému je několikanásobné spuštění algoritmu s různými počátečními řešeními [4] [17].

6.3.2 Horolezecký algoritmus (Hill Climbing Algorithm)

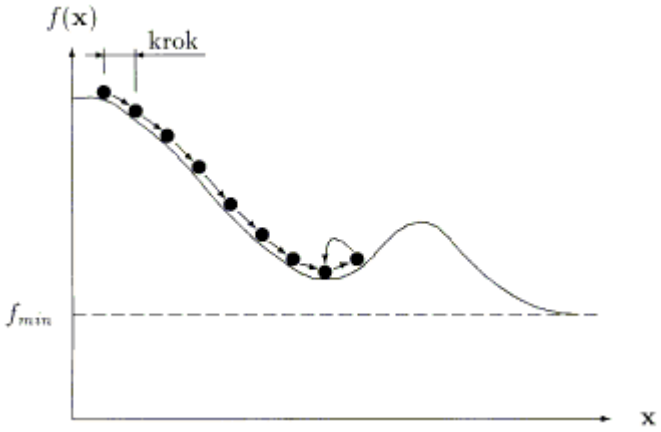
Horolezecký algoritmus pracuje na podobném principu jako metoda lokálního prohledávání. Opět se prohledávají sousední řešení. Tento algoritmus má dva základní rozdíly. První z nich spočívá v tom, že algoritmus umožňuje postup při prohledávání i za předpokladu zhoršení aktuálního řešení. Druhým rozdílem je, že algoritmus je ukončen až po určitém počtu zvolených iterací. V základní formě je stochastičnost tohoto algoritmu pouze ve výběru počátečního řešení.

Postup horolezeckého algoritmu:

1. Zvolíme náhodně počáteční řešení $x_0 \in X$, které je přípustné. Položíme $x_{nej\ lep\ší} = x_0$, $k = 0$ a zvolíme počet iterací k_{max} .
2. Pro všechna sousední řešení $x_{k+1} \in N(x_k)$, kde N je nejbližší okolí a určíme nejnižší hodnotu účelové funkce $f(x_{k+1})$ z okolí N.
3. Pokud nevyhovuje žádný prvek, pak algoritmus končí.
4. Jestliže $f(x_{k+1}) < f(x_{nejlep\ší})$, potom $x_{nejlep\ší} = x_{k+1}$. Zvýšíme iteraci $k = k + 1$ a pokračujeme krokem 2.
5. Pokud $k_{max} = k$, pak je algoritmus ukončen.

Postup prohledávání může vést k tomu, že se algoritmus může opakovaně

vracet do stejného aktuálně nejlepšího optima a tím dochází k zacyklení. Tento algoritmus může mít například úpravu, kdy je v každém kroku vybrána náhodně pouze část sousedních řešení [4] [17].



Obrázek 17 - Zacyklení u horolezeckého algoritmu [4]

6.3.3 Metoda zakázaného prohledávání (Tabu Search)

Základní princip metody pochází z horolezeckého algoritmu. Metoda zakázaného prohledávání se snaží zamezit problému zacyklení. Metoda si pamatuje krátkodobý seznam inverzních změn k předcházejícím změnám, který se nazývá zakázaný seznam (Tabu List). Změny uložené v tomto seznamu jsou při prohledávání okolí aktuálního řešení zakázány. Tímto jednoduchým způsobem metoda odstraňuje problém zacyklení. Velikost zakázaného seznamu je omezena určitým počtem kroku. Tento seznam je na začátku prázdný a vytváří se až za běhu algoritmu. Zakázaný seznam může být implementován například pomocí kruhové fronty. Pokud totiž seznam dosáhne maximální velikosti, pak prvek, který čeká ve frontě nejdéle je z ní odstraněn. Pro zakázaný seznam dále platí pravidlo, že by jeho délka neměla přesahovat počet možných změn v rámci sousedního okolí. Délka zakázaného seznamu má velký vliv na průběh algoritmu. Pokud je tato délka příliš krátká, pak je možné, že dojde k zacyklení algoritmu. Jestliže naopak je seznam příliš dlouhý, může se stát, že přeskočíme některá nadějná řešení.

Algoritmus metody zakázaného prohledávání:

1. Zvolíme počáteční řešení $x_0 \in X$, které je přípustné. Položíme $x_{nejlepší} = x_0$, $k = 0$ a zvolíme maximální délku seznamu n.
2. Pro všechna sousední řešení $x_c \in N(x_k)$, kde N je nejbližší okolí.

Jestliže se změna $x_k \rightarrow x_c$ nachází v zakázaném seznamu, pak opakujeme krok 2.

3. Jestliže změna $x_k \rightarrow x_c$ není zakázána, pak $x_{k+1} = x_c$. Uložíme inverzní změnu do zakázaného seznamu. Pokud je délka seznamu delší nežli n, pak odstraníme nejstarší změnu ze zakázaného seznamu.
4. Jestliže $f(x_c) < f(x_{nejlep\ší})$, pak $x_{nejlep\ší} = x_c$.
5. Pokud je splněno ukončovací kritérium, pak algoritmus končí.
6. Zvýšíme iteraci $k = k + 1$ a pokračujeme krokem 2.

Zakázaný seznam se používá k modifikaci sousedství aktuálního řešení. V tomto sousedství není řešení vzniklé změnami obsaženými v zakázaném seznamu. Výjimka může nastat pouze tehdy, je-li splněno aspirační kritérium. Příkladem může být, že povolíme změnu ze zakázaného seznamu, pokud bychom získali lepší hodnotu účelové funkce, než jsme doposud nalezli.

Koncepce dlouhodobého zakázaného seznamu používá dvě strategie. Při opakování určitých změn v zakázaném seznamu se zaznamenává jejich frekvence. První ze strategií, intenzifikační, podporuje ta řešení, která dosahují lepších výsledků. Zatímco diversifikační přístup upřednostňuje řešení, která se významně lišila od předchozích nalezených řešení [18].

6.3.4 Metoda simulovaného žíhání (Simulated Annealing)

Metoda simulovaného žíhání je inspirována horolezeckým algoritmem. Metoda používá stochastické operátory, které umožňují s určitou pravděpodobností přijmout i horší řešení. Dalším rozdílem je, že se neprohledává sousedství řešení, ale pomocí operátoru se stochasticky transformuje aktuální řešení. Tímto způsobem dojde k prohledání celého prostoru, nikoliv pouze jeho lokální části.

Základní ideou metody je fyzikální děj, který probíhá při žíhání tuhého tělesa. Během procesu žíhání dochází k odstraňování defektů krystalické mřížky. Krystaly tělesa se zahřejí na vysokou teplotu, která se postupně pomalu snižuje. Defekty v krystalické mřížce při vysokých teplotách s velkou pravděpodobností zaniknou. Při postupném ochlazování je menší šance vzniku nových defektů. Výsledkem je dosažení soustavy, kdy jsou všechny atomy v rovnovážných polohách a krystaly tělesa neobsahují žádné vnitřní defekty.

Metropolisovo kritérium pochází z padesátých let, kdy Metropolis použil simulační metodu Monte Carlo pro určení termodynamických konstant plynu. Původní řešení *x* se nahradí novým x' z okolí *x* s pravděpodobností dané Metropolisovým vzorcem:

$$P(x \to x') = \begin{cases} 1 & \text{pro } f(x') \leq \mathrm{f}(x) \\ e^{-\frac{f(x')-f(x)}{T}} & \text{pro } f(x') > \mathrm{f}(x) \end{cases}$$

Parametr *T* je analogií teploty systému. Pokud je hodnota účelové funkce nového řešení stejná nebo lepší, pak pravděpodobnost nahrazení původního řešení je 1. Jestliže nastane druhý případ, kdy hodnota účelové funkce nově vytvořeného řešení je nižší, pak bude aktuální řešení nahrazeno s pravděpodobností $0 < P(x \to x') \leq 1$. Když hodnota parametru *T* nabývá vysokých hodnot, pak je pravděpodobnost přijetí nového řešení vysoká. V opačném případě, kdy parametr *T* se blíží k nule, se nová řešení akceptují jen výjimečně. Tento jev lze vysvětlit tím, že na začátku celého procesu simulovaného žíhání, kdy je teplota vysoká, jsou umožněny velké skoky v prostoru řešení a existuje větší pravděpodobnost, že nová řešení budou přijata i za cenu, že budou horší. S postupným poklesem teploty se snižuje velikost skoků v prohledávání prostoru řešení i pravděpodobnost přijetí nového řešení.

Algoritmus simulovaného žíhání:

1. Zvolíme náhodně počáteční řešení $x_0 \in X$, které je přípustné. Položíme $x_{nejlepší} = x_0$, počet iterací $k = 0$, inicializujeme počáteční teplotu $T_0 = T_{max}$, maximální počet iterací k_{max} a nakonec zvolíme koeficient α redukující teplotu *T*.
2. Vybereme řešení $x' \in N(x_k)$, kde *U* je okolí řešení x_k a určíme hodnotu účelové funkce $f(x')$.
3. Jestliže $f(x') \leq f(x_{nejlepší})$, potom $x_{nejlepší} = x'$ a $x_{k+1} = x'$. Pokračujeme krokem 6.
4. Pokud $f(x') > f(x_k)$, pak $x_{k+1} = x'$ a pokračujeme krokem 6.
5. Vygenerujeme náhodné číslo $0 < U < 1$. Jestliže $U < e^{-\frac{f(x')-f(x_k)}{T_k}}$, pak $x_{k+1} = x'$, jinak $x_{k+1} = x_k$.
6. Aktualizujeme teplotu $T_{k+1} = \alpha \cdot T_k$ a zvýšíme iteraci $k = k + 1$.

7. Jestliže je aktuální teplota $T_k < T_{min}$, tak algoritmus končí, jinak pokračujeme krokem 2.

Koeficient α udává způsob, jakým se bude snižovat teplota T. V praxi se hodnota tohoto koeficientu volí v rozmezí 0,8 až 0,99. Existují sofistikovanější verze tohoto algoritmu, kde lze koeficient α během procesu simulovaného žíhání měnit. Jednou z variant je využití genetického algoritmu pro řízení teploty [16].

6.3.5 Genetické algoritmy

Genetický algoritmus je heuristický postup využívající principy evoluční biologie a představuje zvláštní případ stochastických algoritmů. Genetické algoritmy byly původně navrženy pro evoluční procesy. Z tohoto důvodu je terminologie podobná terminologii používané v biologii.

Genetické algoritmy se odlišují od klasických optimalizačních metod univerzalitou a širokým záběrem. Tyto algoritmy pracují s globální strukturou nazývanou chromosom, který nese zakódované parametry. Pro postup hledání optima pak vyžadují pouze účelovou funkci.

Základem je vektor popisující jeden, či více parametrů, představující chromosom. Chromosomy se skládají z genů představující jednotlivé bity. Každý gen představuje určitý typ vlastnosti. Dalším pojmem je alela představující hodnotu, kterou může gen nabývat. Genetická informace se nazývá genom. Nejvíce se využívají binární chromosomy, které jsou reprezentovány hodnotami 0 nebo 1. Množina chromosomů, či jedinců představuje populaci. Každý chromosom populace má hodnotu zdatnosti fitness, která je určena účelovou funkcí.

Genetické algoritmy využívají tři genetické operace. Během operace reprodukce dochází ke kopírování chromosomů do další generace. Pravděpodobnost přechodu je určena hodnotou fitness. Čím vyšší je tato hodnota, tím větší je šance pro přežití. Operace křížení rekombinuje vybrané páry chromosomů. Posledním procesem je operace mutace, která s velmi malou pravděpodobností změní hodnotu alely. Tím se zabraňuje ztrátě genetické informace ve špatných chromosomech.

Princip genetických algoritmů spočívá ve vytvoření nulté generace, která se mění pomocí operátorů selekce, křížení a mutace. Při selekci dojde k výběrů párů podle vybraného selekčního schématu. Vybíráme jedince s vyšší hodnotou fitness,

ale nemůžeme preferovat pouze ty nejlepší. Způsobilo by to konvergenci k lokálnímu optimu. Jedním ze selekčních schémat je například turnajové schéma, které vybere náhodně jedince a z nich vybere nejlepšího. Velmi často jsou vybíráni jedinci do turnaje po dvou. Dalšími příklady schémat jsou selekce pořadím, ruletové kolo, Sigma Scalling, Boltzmann Scalling a další. Po křížení může nastat s malou pravděpodobností mutace, která snižuje možnost uváznutí v lokálním extrému. Algoritmus pak končí splněním ukončovací podmínky.

Princip genetického algoritmu:

1. Zvolíme náhodnou populaci jedinců $x_0 \in X$, která je přípustná. Pokud optimalizační úloha obsahuje *v* rozhodovacích proměnných, které jsou zakódovány pomocí *n* binárních míst, pak je délka chromosomu dána vztahem $L = v \cdot n$, kde L je počet bitů.
2. Vypočítáme hodnoty fitness, tj. účelových funkcí všech jedinců populace.
3. Pokud je splněno ukončovací kritérium, pak je algoritmus ukončen.
4. Vytvoření nové populace:

   4a. krok: Selekce - náhodně vybereme dva rodičovské jedince za pomoci selekčních schémat. Čím vyšší hodnota účelové funkce jedince, tím vzniká větší pravděpodobnost, že bude vybrán.

   4b. krok: Křížení – pomocí rekombinace páru rodičů vytvoříme dva nové jedince. Nejjednodušší z metod je například jednobodové křížení, kdy určíme náhodným způsobem bod překřížení *k*. První jedinec je pak vytvořen pomocí genů jednoho ze svých rodičů na pozicích 1 až k a geny druhého rodiče na pozicích $k + 1$ až L, kde L je délka chromosomu. Druhý jedinec získá geny opačné části genů rodičů.
5. Mutace – se provádí na každém jedinci jednotlivě. S velmi malou pravděpodobností dojde ke změně genetické informace. Zpravidla se volí $P = 0{,}0001$.
6. Pokračujeme krokem 2.

Jako ukončovací kritérium lze zvolit počet iterací. Tento způsob ovšem není příliš vhodný. Při velkém počtu iterací bychom ztráceli čas zbytečným křížením. V opačném případě se může stát, že populace nestihne zkonvergovat. Efektivnější je varianta ukončení, když se hodnota účelové funkce po několik iterací nezlepšuje. Počáteční populace je rozprostřena do celého prostoru řešení. Postupně se začnou vytvářet shluky jedinců kolem optimálních řešení. Pokud se shluky budou k sobě přibližovat, pak má hledaný problém jedno optimální řešení, nebo naopak, pokud se shluky budou od sebe oddalovat, pak existuje pro úlohu více řešení stejné hodnoty [4] [16].

6.4 Porovnání vybraných metod

Exaktní metoda větví a mezí je jedinou z uvedených metod, která najde se stoprocentní jistotou optimální řešení. Metoda se nedá použít pro rozsáhlé projekty, jelikož prohledává velkou část prostoru řešení a její výpočetní složitost roste exponenciálně.

Metoda zakázaného prohledávání může být použita ve spojení s horolezeckým algoritmem, tak i v kombinaci s jinými algoritmy jakými jsou metoda simulovaného žíhání nebo genetické algoritmy. S těmito metodami však nebývá příliš efektivní, jelikož tyto metody neprohledávají celé okolí aktuálního řešení. Z tohoto důvodu je účinnost zakázaného seznamu velmi malá.

Metoda simulovaného žíhání byla testována už v době svého vzniku na známém problému obchodního cestujícího, kdy hledáme nejkratší uzavřenou cestu mezi *N* místy. Tento problém patří mezi NP-úplné problémy. Čas nalezení řešení roste při zvětšování dimenze v nejlepším případě exponenciálně. Simulované žíhání dokáže nalézt optimální řešení (u složitějších úloh většinou však jen suboptimální). Ukazuje se, že simulované žíhání poskytuje velmi efektivní algoritmus k řešení kombinatorických úloh, které jsou NP-úplné, přičemž získaná řešení jsou buď totožná nebo velmi blízká optimálnímu řešení.

Genetické algoritmy překvapivě dobře fungují při řešení problémů, kde téměř všechny ostatní algoritmy selhávají, např. pro NP-úplné problémy, tj. kde výpočetní čas je exponenciálně nebo faktoriálně závislý na počtu proměnných. Nemá však smysl je používat u relativně jednoduchých optimalizovaných funkcí nebo u funkcí, pro které existují specializované algoritmy [19].

7 Formulace řešené problematiky

Hlavním úkolem bylo vytvoření softwarového nástroje pro řízení rozsáhlých projektů pomocí síťové analýzy. Cílem této práce je rovněž vyřešení problematiky pochopení základních principů algoritmů řízení projektů běžnému uživateli.

Pro řízení projektů pomocí síťové analýzy jsou stěžejními metodami CPM a PERT, jejichž rozdíl je zejména v pohledu na časové trvání jednotlivých činností. Tato problematika je poměrně podrobným způsobem popsána v kapitole 4.

Tyto metody ve své čisté podobě se nezabývají problémem rozmístění zdrojů (pracovních sil, výrobních zařízení, spotřebních materiálů aj.), avšak jsou důležitým podkladem pro jejich řešení. Úloha rozmístění zdrojů patří do skupiny NP-těžkých úloh, kdy je potřeba prozkoumat velké množství variant rozmístění zdrojů. Takový problém není možné řešit zkoumáním všech existujících řešení. Z tohoto důvodu byly popsány v předchozí kapitole vybrané metody, které redukují počet zkoumaných řešení, a tedy vedou ke zkrácení výpočetního času.

Pro výukový program bylo zvoleno řešení rozmístění zdrojů v síťovém grafu pomocí metody větví a mezí. Algoritmus této metody dokáže poměrně jednoduchým a přirozeným způsobem poukázat na princip řešení daného problému, což je velmi důležité pro jeho pochopení. Nelze ovšem opomenout velkou nevýhodu metody větví a mezí při rozsáhlejších projektech. Důvodem je, že se jedná o exaktní metodu, kde algoritmus prochází velké množství řešení, jejichž počet roste exponenciálně. Proto byl v práci tento algoritmus částečně upraven a snaží se prohledávat pouze ta řešení, tak aby v každém kroku byly jednotlivé zdroje maximálním způsobem zatíženy, nebo bylo zpracováno co největší možné množství činností současně. Důsledkem této úpravy je prastrom řešení významným způsobem redukován. S tím ovšem dochází k vedlejšímu efektu, že nebudou nalezena všechna optimální řešení.

7.1 Algoritmus pro rozmístění zdrojů v síti

Popis algoritmu pro rozmístění zdrojů v síťovém grafu si popíšeme pomocí metody větví a mezí při nedostatečné kapacitě zdrojů. Tedy úkolem je vykonat posloupnost činností v takovém sledu, aby respektovaly kapacitní omezení a celková doba trvání projektu byla minimální.

V teoretickém příkladu budeme uvažovat síťový graf s deterministickým

ohodnocením doby trvání činnosti. Jednotlivé činnosti lze rozlišit na dva případy. Kterákoliv započatá činnost může být kdykoliv pozastavena a její zdroje mohou být uvolněny pro vykonávání ostatních libovolných činností. Druhou variantou jsou činnosti, které nelze přerušit. Tyto činnosti musí být po rozpracování dokončeny bez přerušení. V práci se budeme zabývat pouze variantou, že libovolnou činnost lze kdykoliv přerušit a její zdroje tak použít pro zbylé činnosti projektu. Ideálním řešením by ovšem bylo rozlišovat jednotlivé typy činností na přerušitelné a nepřerušitelné.

Z problematiky, pokud lze libovolné činnosti pozastavit, lze uvažovat následující strategie:

- Při výběru činností, které mohou být zpracovávány souběžně, je upřednostňována varianta, aby příslušné činnosti nemusely být přerušeny.
- Vyšší prioritu bude mít dokončení pozastavené činnosti před zpracováním činnosti zatím nezapočaté.

Uvedené strategie můžeme kombinovat. V této práci zatím není upřednostňována ani jedna z variant. Jedná se pouze o návrhy, které by mohly být užitečným nástrojem pro prohledávání možných řešení úlohy.

K dispozici máme zdroje $z_1, \dots, z_n$ o příslušných kapacitách $k_1, \dots, k_n$. Zdroje jsou rozlišovány na neobnovitelné (spotřební materiál, elektrická energie, voda, apod.) a obnovitelné zdroje představující pracovní síly (pracovníci, výrobní zařízení). Neobnovitelné zdroje po použití jsou nenávratně spotřebovány a už je nelze znovu použít. U těchto zdrojů má smysl z hlediska síťové analýzy sledovat pouze jejich spotřebované množství pro dokončení všech svých činností. Při nedostatku neobnovitelných zdrojů pro splnění prací má za následek nedokončení celého projektu a tedy úlohu lze pak označit za neřešitelnou.

Naopak obnovitelné zdroje jsou po dokončení zadané práce opět uvolněny. Jediným požadavkem na obnovitelné zdroje pro dokončení projektu je jejich dostatek pro zpracování kterékoliv činnosti samostatně. V případě, kdy by bylo možné zpracovat n činností bez ohledu na dostatečné množství obnovitelných zdrojů, zatímco při kapacitním omezení počet těchto činností by byl menší nežli n, pak má smysl sle-

dovat, jakým způsobem dané zdroje v příslušném síťovém grafu využít. To znamená, že pokud bychom měli k dispozici 2 zaměstnance a mohli bychom vykonávat dvě činnosti současně a jedna z nich potřebovala pro své dokončení 2 zaměstnance a druhá jednoho, pak nelze tyto práce vykonávat současně. Významnou roli zde hraje sledování vytížení všech obnovitelných zdrojů během celého projektu a dále práce s touto statistikou pro optimalizaci projektového plánu.

Účelovou funkci představuje algoritmus metody kritické cesty pro síťový graf bez kapacitního omezení, jelikož se snažíme v každém kroku minimalizovat čas trvání celého projektu.

Jednotlivé uzly prastromu představují řešení příslušného podproblému. Hrany prastromu řešení představují vykonané, nebo z části vykonané, činnosti síťového grafu. Hrana prastromu může představovat i několik činností současně. Další informaci, kterou hrana prastromu nese je čas informující, kolik časových jednotek bylo spotřebováno pro vykonání činností příslušných hraně. Zpětným průchodem z libovolného listu prastromu směrem ke kořeni získáme přehled o doposud vykonaných činnostech.

Postup vytváření prastromu řešení při řešení úlohy s kapacitním omezením:

1. krok: Definujeme zdroje projektu $z_1, \dots, z_p$ o příslušných kapacitách $k_1, \dots, k_p$, kde p je počet různých typů zdrojů. $x_{nejlep\ ší} = \infty$. Inicializujeme prastrom řešení, do kterého vložíme kořen, který představuje počáteční stav řešení úlohy. Kořen prastromu odpovídá počátečnímu uzlu síťového grafu. Vypočítáme hodnotu účelové funkce kořene prastromu, což je spodní hranice. Tj. určíme hodnotu kritické cesty celého projektu bez kapacitního omezení. $b_0 = CPM(P_0)$. Nově vytvořený uzel vložíme do množiny neprozkoumaných řešení $E = \{P_0\}$.

2. krok: Je-li množina $E = \emptyset$, pak algoritmus končí.

3. krok: Vybereme uzel P, který je listem prastromu řešení a odebereme jej z množiny neprozkoumaných řešení $E = E \setminus \{P\}$.

4. krok: Zpětným průchodem prastromem řešení z uzlu P ke kořeni P_0 zjistíme, které činnosti, respektive části činností, které byly zpracovány. Dokončené činnosti označíme jako vykonané a nebudeme je brát

v potaz při řešení. Dá se říci, že vykonané činnosti vypustíme z původního síťového grafu. Tímto nám vznikne podgraf představující jeden konkrétní podproblém úlohy. V případě, že činnost byla vykonána částečně, pak upravíme v síťovém grafu její časové ohodnocení odečtením vykonané práce na činnosti od její celkové doby trvání.

5. krok: Určíme činnosti síťového grafu, které mají všechny předcházející činnosti vykonané a vložíme je do posloupnosti *N*. Z těchto činností vytvoříme všechny možné kombinace respektující kapacitní omezení zdrojů. Tzn. že pro každou kombinaci činností určíme jednotlivé sumy zdrojů

$$s_j = \sum_{h \in N} z_j[h], \text{ kde } j = 1, \dots, p.$$

6. krok: Všechny součty s_j porovnáme s kapacitami příslušných zdrojů k_j, kde $j = 1, \dots, p$. Pokud je $s_j \leq k_j$, pak příslušnou kombinaci činností lze vykonat a vložíme ji do posloupnosti *K*, v opačném případě kombinaci zamítneme.

7. krok: Pro všechny kombinace posloupnosti *K* vložíme do prastromu řešení uzel P_i, představující konkrétní stav řešení úlohy. Z uzlu P do uzlu P_i vedeme hranu $h = [P, P_i]$, která odpovídá příslušné kombinací činností. Hrana má ohodnocení rovné nejkratší z činností kombinace. Pro nově vytvořené uzly P_i vypočítáme hodnotu spodní hranice jako součet kritické cesty podproblému a délky cesty z aktuálního uzlu ke kořeni:

$$b_i = CPM(P_i) + |m(P_i, P_0)|.$$

7a. krok: Jestliže $CPM(P_i) = 0 \wedge b_i < f(x_{nejlep\ší})$, potom přiřadíme $x_{nejlep\ší} = P_i$.

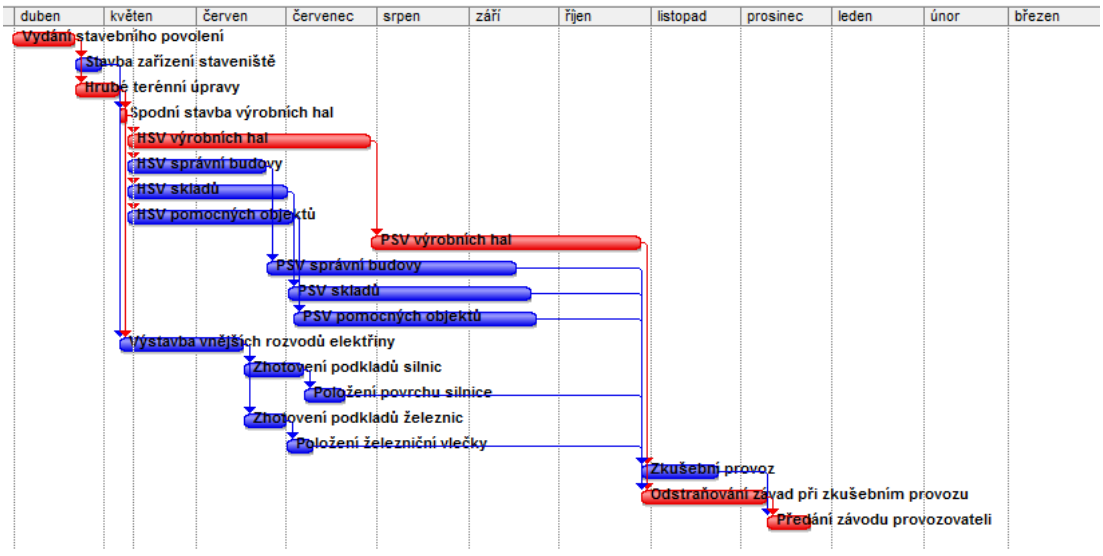
7b. krok: Jestliže $CPM(P_i) = 0 \wedge b_i \geq f(x_{nejlep\ší})$, pak řešení zamítneme. Jinak $E = E \cup \{P_i\}$.

8. krok: Návrat do kroku 2.

8 Přehled vybraných SW produktů pro řízení projektů

V dnešní době je na trhu k dispozici několik desítek softwarových nástrojů pro řízení a plánování projektů. Produkty se liší rozsahem množností náhledu na projekt, grafickým zpracováním i cenou.

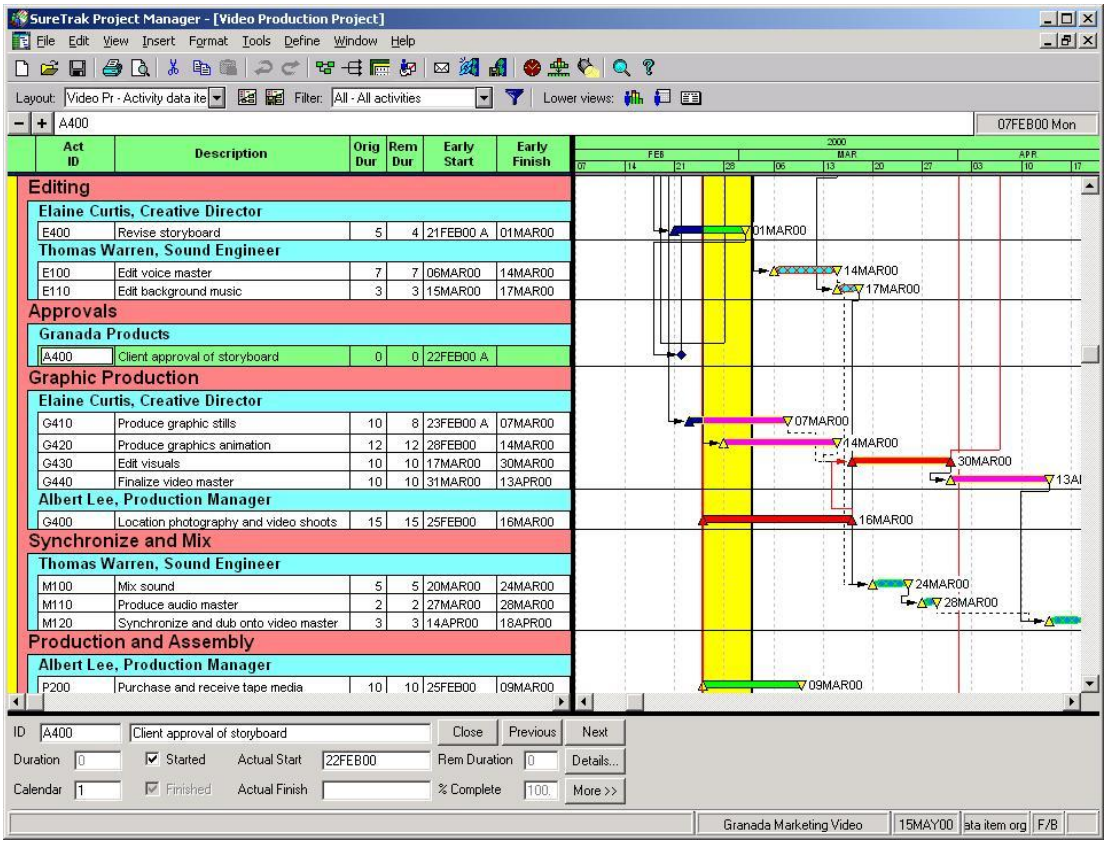
Microsoft Office Project 2007 je jedním z nejvíce používaných nástrojů poslední doby. Ve verzi Standard nabízí celou řadu nástrojů pro řízení a plánování projektů. Aplikace obsahuje podrobného průvodce projektem, součástí kterého je podrobný interaktivní návod, který uživateli umožní rychlé vytvoření projektu, přiřazení úkolů a zdrojů, analýzu dat nebo sledování projektu. Grafické znázornění projektu je v aplikaci možné v několika formách. Nejvíce využívanou formou jsou Ganttovy diagramy, síťový graf, či sledovací Ganntův diagram. Aplikace umožňuje pohled na dobu trvání činnosti jak deterministickým (metoda CPM), tak i stochastickým způsobem (metoda PERT). Velkou výhodou aplikace Office Project Standard 2007 je integrovaná podpora komunikace s ostatními programy sady Microsoft Office. Součástí aplikace je i propracovaná nápověda s podrobnými návody pro tvorbu projektových plánů. Mezi další služby patří online přístup k šablonám, školicím kurzům a dalším informacím.



Obrázek 18 - Ganttův diagramu v aplikaci MS Office Project 2007 – Zdroj: vlastní

Jedním z vůdčích poskytovatelů na trhu je také firma Primavera Software, Inc., která byla převzata společností Oracle. Firma nabízí velké množství produktů zaměřených na řízení projektů. Jedním z nich, **Primavera Project Planner**, je nástroj pro řízení rozsáhlých projektů. Aplikace pracuje s Ganttovy diagramy, PERT

grafy a diagramy založených na časovém měřítku. Primavera Project Planner umožňuje také řízení zdrojů (lidé, peníze, materiál, čas). Pro menší a střední projekty firma Primavera vyvinula software pod názvem **SureTrak Project Manager**. Jedná se o jednoduchý, ale efektivní software pro plánování a řízení projektů. Pro grafické znázornění jsou k dispozici Ganttův diagram nebo PERT graf. Aplikace je velmi pěkně zpracována a do detailu promyšlena. Mezi klady patří přehledné sledování zdrojů, nákladů a možnost vytváření rozsáhlých reportů. Nevýhodou aplikace je chybějící lokalizace a čeština obecně (názvy a jména nepodporují diakritiku apod.).



Obrázek 19 - SureTrak Project Manager – Zdroj[2]

Obě aplikace patří mezi profesionální produkty. Mají intuitivní editaci projektu a velmi přehledně zobrazují výsledky. V těchto aplikacích však chybí podrobnější popis základních metod, které využívají k řešení řízení projektu. Neobeznámí tak uživatele se základy dané problematiky, což může vést ke snížení kvality řízení pro-

[2] *SureTrak Project Manager od firmy Primavera* [online]. [2006] [cit. 2009-05-13]. Dostupný z WWW: <http://rizeni-projektu.cz/view.php?cisloclanku=2006052901>.

jektu. Aplikace v této práci zadanou problematiku řeší podrobným popisem postupu algoritmu příslušných metod a osvětluje uživateli základní principy při tvorbě síťového grafu, určení kritických činností metodou kritické cesty, či dodržení plánového termínu metodou PERT a v neposlední řadě problematiku rozmístění kapacitně omezených zdrojů v projektu.

Na závěr této kapitoly si uvedeme tabulky některých dostupných softwarových nástrojů pro řízení a plánování projektů. První tabulka obsahuje freeware a open-source aplikace.

Název aplikace	Typ aplikace	Platforma	Výrobce
Bugzilla	Web	Multiplatformní	Mozzila Foundation
Codendi	Web	Multiplatformní	XEROX
Collabtive	Web	Multiplatformní	Philipp Kiszka
eGroupWare	Web	Multiplatformní	Open Source Project
Gantt Project	Desktop	Linux, Windows, Mac OS X	The GanttProject Team
Mantis BugTracker	Web	Multiplatformní	Victor Boctor
OpenProj	Desktop	Multiplatformní	Projity
Open Workbench	Desktop	Windows	Niku
TaskJuggler	Desktop	Linux	The TaskJuggler Team
OpenGoo	Web	Multiplatformní	Feng Office
ProjectPier	Web	Multiplatformní	ProjectPier.org team
Project.net	Web	Multiplatformní	Project.net
Track	Web	Multiplatformní	Edgewall software

Tabulka 4 - Přehled open-source a freeware aplikací – Zdroj [20]

V druhé tabulce jsou uvedeny placené webové a desktopové aplikace pro řízení a plánování projektů. Jejich cena se pohybuje od stovek, až po statisíce korun.

Název aplikace	Typ aplikace	Platforma	Výrobce
Artifact Software	Web	Multiplatformní	Privately-held
Blue Ant	Web	Multiplatformní	Proventis
CAMeLEAN/PM	Web	Linux, Windows	Ranal Software Solutions
Cardinis	Web	Multiplatformní	Cardinis Solutions
EPM live	Web	Multiplatformní	Privately-held
LiquidPlanner	Web	Multiplatformní	Privatly-held
MacProject	Desktop	Mac OS X	Solosoft, Claris
MicroPlanner X-PERT	Desktop	Windows	Micro Planning International
Microsoft Office Project	Desktop	Windows	Microsoft
Microsoft Office	Web	Windows	Microsoft

Project Server			
OmniPlan	Desktop	Mac OS X	The Omni Group
P2ware Planner	Desktop	Windows	P2ware
Planisware5	Desktop	Multiplatformní	Planisware
Projekt Kaiser	Web	Multiplatformní	Triniforce
Projektron BCS	Web	Multiplatformní	Projektron
Primavera Project Plannet	Desktop	Windows	Primavera
Project KickStart	Desktop	Windows	Experience in soft-ware, Inc.
Rational Plan	Desktop	Linux, Windows, Mac OS X	Stand By Soft
RiskyProject	Desktop	Windows	Intaver Institut
SureTrak Project Manager	Desktop	Windows	Primavera
Tracker	Desktop	Mulitplatformní	Automation Centre
Viewpath	Web	Multiplatformní	Privately-held

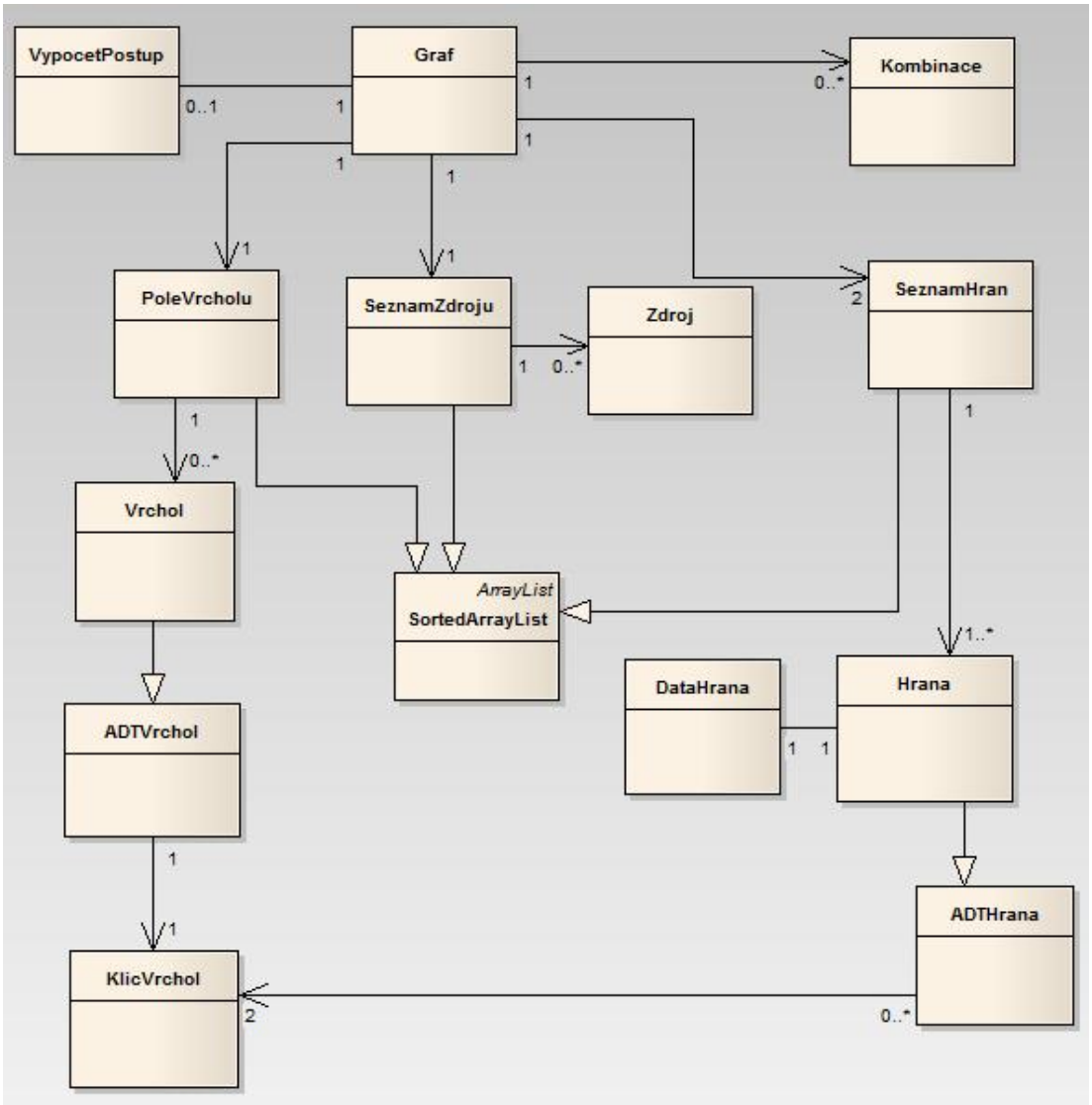
Tabulka 5 - Přehled komerčních aplikací - Zdroj [20]

9 Softwarové řešení

Praktická část diplomové práce spočívá v implementaci výukového nástroje pro řízení rozsáhlých projektů pomocí síťové analýzy, která usnadní uživateli lepší pochopení dané problematiky. Aplikace pracuje s deterministickým i stochastickým chováním doby trvání činnosti.

9.1 Návrhový model aplikace

V této podkapitole zmíníme pouze vybrané třídy návrhu. Celkový strukturální pohled na návrh včetně kardinalit mezi třídami je uveden na následujícím obrázku. Základní třídou návrhu je *Graf*. V této třídě se nachází téměř všechny algoritmy a výpočty.



Obrázek 20 - Strukturální návrh tříd – Zdroj: vlastní

9.2 Popis základních tříd

- ***Graf*** – třída, která obsahuje, včetně základních operací s grafem, téměř všechny algoritmy pro práci s řízením projektů. Tato třída ukládá data v paměti do datové struktury typu dopředně-zpětná hvězda, která je popsána v následující kapitole. Uvedeme si přehled základních metod třídy *Graf*:
    - ***AlgoritmusTopologickehoUsporadani*** – metoda, která ohodnotí všechny uzly síťového grafu algoritmem maximální dráhy. Z důvodu, že pracuje se síťovým grafem, metoda ověřuje některá základní pravidla tvorby síťových grafů. Metoda si určí počáteční a koncový uzel sama. Testuje existenci právě jednoho počátečního a koncového uzlu síťového grafu, případně zdali takový uzel vůbec existuje. V neposlední řadě metoda řeší problém acykličnosti grafu. Tento problém je více rozebrán přímo v algoritmu topologického uspořádání (viz. kapitola 9.4.1).
    - ***VypocetRezerv*** – v této metodě se vypočítají jednotlivé rezervy všech činností. Jedná se o rezervu celkovou, volnou a nezávislou.
    - ***AlgoritmusBaB*** – je metoda obsahující algoritmus pro rozmístění kapacitně omezených zdrojů v síťovém grafu. Využívá několika metod, z nichž si uvedeme pouze tři stěžejní. První z nich ***VytvorGrafPodproblemu*** transformuje původní síťový graf na jeho podgraf odpovídající danému podproblému. Druhou metodou je ***VytvorNovaSubreseni***, ve které se procházejí všechny přípustné kombinace a provádí se výpočty účelových funkcí metodou kritické cesty. Třetí z významných metod je ***ziskatIndexListuNejnizsihoCasu***, která vybírá dle zvolené strategie uzel s nejnižší hodnotou účelové funkce.
    - ***VypocitejSouradnice*** – určí souřadnice jednotlivých uzlů v grafu. Tento algoritmus není kompletně vyřešen. Není zde řešena možnost, kdy uzel splývá s hranou, se kterou neinciduje. Tento problém by se dal odstranit například pomocí výpočtu jednotlivých směrnic hran, tak aby žádný uzel, který není počátečním ani koncovým uzlem hrany, neležel na této hraně. Řešení zobrazení grafu záleží pak na interakci uživatele, který může měnit polohu libovolného uzlu.
- ***Hrana*** – třída obsahující informace o hraně. Každá hrana musí obsahovat identi-

fikaci pomocí dvou incidujících uzlů (třída ***KlicVrcholu***). Vlastní informace hrany jsou pak reprezentovány třídou ***DataHrana***.

- ***DataHrana*** – obsahuje veškerá potřebná data hrany a metody pro práci s nimi. Každá hrana obsahuje informace o časovém ohodnocení, rezervách, názvu, potřebných zdrojích pro její vykonání, apod.
- ***Zdroj*** - obsahuje informace o zdroji, jeho popis, maximální kapacitu, aktuální kapacitu, obnovitelnost zdroje a v neposlední řadě také finanční ohodnocení.
- ***Vrchol*** – obsahuje atributy pro určení polohy uzlu na obrazovce, názvu a lhůtových ukazatelů. Jsou zde obsaženy metody pro ukládání a načítaní uzlu ze souboru.
- ***PostupVypoctu*** – tato třída se stará o zpracování textového záznamu o průběhu jednotlivých algoritmů metod CPM, PERT a také algoritmu pro rozmístění kapacitně omezených zdrojů metodou větví a mezí.
- ***SortedArrayList*** – třída odvozená od kolekce *ArrayList*, což je indexovaná struktura. Tato kolekce by se dala charakterizovat jako „nafukovcí“ pole. Při nedostatečné kapacitě pole si kolekce sama alokuje další potřebnou paměť. Každá alokace ovšem zpomalí chod celé aplikace. Při vytváření instance této třídy ovšem můžeme definovat počet alokovaných prvků pole. Třída *SortedArrayList* je od třídy *SortedArrayList* odvozena a implementuje binární třídění.
- ***PoleVrcholu*** – je třída odvozená od třídy *SortedArrayList*. Obsahuje všechny uzly síťového grafu. Implementuje metody pro ukládání a načítaní uzlů při práci se soubory a implementuje rozhraní *IComparer* pro porovnávání uzlů. Další důležitou funkcí této třídy je binární vyhledávání uzlů grafu.
- ***SeznamHran*** – třída odvozená od třídy *SortedArrayList*. Obsahuje všechny uzly síťového grafu. Struktura *Graf* obsahuje tuto strukturu dvakrát. Jedna slouží pro uchování předchůdců a druhá pro uchování následníků. Ostatní metody jsou analogií k třídě *PoleVrcholu*.
- ***SeznamZdroju*** – třída odvozená od třídy *SortedArrayList*. Obsahuje zdroje projektu. Ostatní metody jsou analogií k třídě *PoleVrcholu*.
- ***Kombinace*** – určí všechny možné kombinace hran bez ohledu na kapacitní ome-

zení zdrojů. Zamítání jednotlivých kombinací provádí metoda *AlgoritmusBaB* třídy *Graf*.

9.3 Reprezentace síťového grafu v paměti

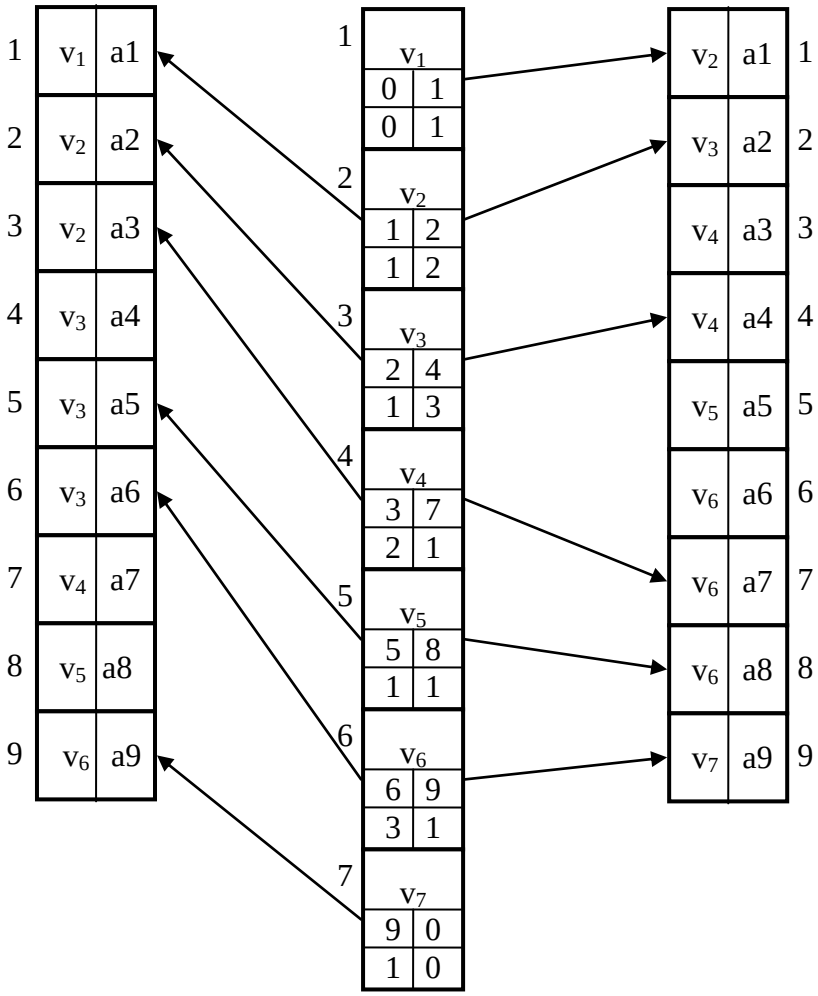
Síťové grafy pro zobrazení projektů nabývají obrovských rozměrů. S tímto problémem nastává otázka, jakou zvolit datovou strukturu pro reprezentaci grafu v paměti, aby bylo garantováno rychlé zpracování jednotlivých operací s prvky grafu a zároveň reprezentace nevykazovala příliš velkou paměťovou náročnost. Datová struktura musí respektovat vlastnosti síťového grafu. Tedy je nutné, aby reprezentace síťového grafu zohledňovala orientaci hran. Z hlediska výpočtů algoritmů nad síťovým grafem musí datová struktura umožňovat pohyb oběma směry. To znamená, že pro libovolný uzel můžeme zpřístupnit následníka i předchůdce.

V této práci byla zvolena grafová reprezentace typu dopředně-zpětné hvězdy. Jedná se o vrcholově orientovanou strukturu, což znamená, že prvotní strukturou pro přístup k jednotlivým prvkům jsou uzly grafu, které zprostředkovávají další informace o jejich předchůdcích a následnících, čímž umožňují přístup k prvkům druhotné struktury reprezentující hrany grafu.

Prvotní struktura je reprezentována pomocí utříděného pole, nad kterým je prováděno binární vyhledávání. Druhotné struktury, představující incidenční uzly, respektive hrany, jsou dvě. Jedná se o dvě tabulky - následníci a předchůdci, které jsou reprezentovány indexovaným seznamem. Popis případné možné implementace dopředně zpětné hvězdy si popíšeme příkladem. Zdrojová data pro příklad jsou uvedena v tabulce 6 a grafická reprezentace vzorového síťového grafu v paměti je znázorněna na obrázku 21.

Činnost	Uzly	Předchůdce	Doba trvání
A	1-2	-	5
B	2-3	A	10
C	2-4	A	6
D	3-5	B	10
E	3-6	B	3
F	4-6	B, C	4
G	6-7	D, E, F	15

Tabulka 6 - Zdrojová data pro příklad dopředně-zpětné hvězdy – Zdroj: vlastní



a1	a2	a3	a4	a5	a6	a7	a8	a9
A	B	C	fikt. hr.	D	E	F	fikt. hr.	G
5	10	6	0	10	3	4	0	15

Obrázek 21 - Příklad dopředně-zpětné hvězdy Zdroj: vlastní

Data prvotní struktury nesou informace o identifikaci uzlu, ve druhém řádku jsou indexy odkazující se do druhotných struktur předchůdců a následníků. Třetí řádek obsahuje informaci o počtu předchůdců a následníků příslušného uzlu. Položka druhotné struktury obsahuje klíč uzlu, který tvoří spolu s uzlem odkazujícího se na tuto položku hranu grafu a ukazatel na adresu v paměti, kde jsou uložena data příslušné hrany. Zjištění předchůdců uzlu je dáno indexem i až indexem $i + p$, kde p je počet předchůdců uzlu. Analogickým způsobem určíme následníky uzlu.

Např. uzel v_6 se odkazuje na index 6 struktury předchůdců a počet předchůdců je 3. To znamená, že uzel v_6 má předchůdce na indexech 6, 7 a 8. Na těchto in-

dexech se ve struktuře předchůdců nacházejí klíče v_3, v_4, v_5 a příslušné ukazatele na paměťová místa $a6, a7, a8$. Bezprostředně předchozí hrany uzlu v_6 jsou tedy $[v_5, v_6], [v_4, v_6], [v_3, v_6]$, což představuje hrany E, F a fiktivní hranu o délkách trvání činnosti 3, 4 a 0.

9.4 Použité algoritmy

Algoritmus metod CPM a PERT je založen na ohodnocení všech uzlů v síťovém grafu. Z čehož vyplývá, že je potřeba použít algoritmus, který určí hodnoty všech příslušných uzlů. Vhodnou volbou pro výpočet lhůtových ukazatelů jednotlivých uzlů síťového grafu je algoritmus topologického uspořádání.

9.4.1 Algoritmus topologického uspořádání

Pokud potřebujeme vykonat nějakou množinu činností takovou, že všechny činnosti musíme vykonat jednu po druhé, tj. nesmíme dělat víc činností najednou a vykonávání žádné činnosti nesmí být přerušeno činností jinou. Tento algoritmus lze použít pouze na acyklických grafech, což je jedna z vlastností síťového grafu.

Pro tento algoritmus si vytvoříme dvě posloupnosti. Jednu pro hrany a druhou bude tvořit posloupnost vrcholů.

Jako první vrchol vybereme vrchol v_1, do kterého nevede žádná hrana a všechny hrany z vrcholu z něj vycházející odstraníme a vložíme je do příslušných posloupností. Zbývající graf je opět acyklický. Tedy existuje opět vrchol, do kterého nevede žádná z hran, tento vrchol označíme v_2. Tento vrchol a všechny hrany, pro které je počátečním vrcholem, z grafu odstraníme a zařadíme je do posloupností. Tento postup opakujeme, dokud nejsou uspořádány všechny vrcholy a hrany. Pokud bychom nechtěli během výpočtu měnit graf, lze se na daný postup dívat i tak, že vybíráme pouze vrchol, jehož předchůdci jsou již v posloupnosti zařazeni. Dále lze algoritmus ještě upravit, tak, že pro každý vrchol grafu v_i udržujeme číslo $C(v_i)$ rovné počtu hran vcházejících do vrcholu v_i z vrcholů, které ještě nejsou zařazeny v posloupnosti. Tyto hodnoty postupně snižujeme a vrcholy, které mají $C(v_i) = 0$ ukládáme do seznamu M jako kandidáty pro zařazení do posloupnosti.

Algoritmus topologického uspořádání:

1. krok: Pro všechny vrcholy $v_i \in V(G)$ položíme $C(v_i) = 0$. Pro všechny koncové vrcholy Kv(h) hran $h \in Y(G)$ provedeme

$C(\mathrm{Kv}(h)) = C(\mathrm{Kv}(h)) + 1$. Pro všechny vrcholy v_i, kde $i \neq 0$ položíme ohodnocení vrcholu určující maximální délku času $U(v_i) = -\infty$. Pro počáteční vrchol tuto hodnotu nastavíme na $v_0 = 0$.

2. krok: Do množiny *M* vložíme všechny vrcholy $v\ v_i$, pro které platí $C(v_i) = 0$.

3. krok: Je-li $M = \emptyset$, pak výpočet končí. Je-li $M \neq \emptyset$, odstraníme z *M* některý vrchol, označíme jej v_i a zařadíme jej na konec posloupnosti. Vrcholů.

4. krok: Pro každou hranu $h \in H^+(v_i)$ provedeme krok 4a a pak pokračujeme krokem 3.

4a. krok: Zařadíme hranu *h* na konec posloupnosti hran, označíme $y = \mathrm{Kv}(h)$, kde Kv je koncový vrchol hrany a položíme $C(y) = C(y) - 1$. Jestliže nyní platí $C(y) = 0$, pak vrchol *y* zařadíme do seznamu *M*. Jestliže ohodnocení koncového vrcholu $U(\mathrm{Kv}(h))$ hrany *h* menší než ohodnocení počáteční vrcholu $U(\mathrm{Pv}(h))$ hrany *h* a jejím ohodnocením, tedy $U(\mathrm{Kv}(h)) < U(\mathrm{Pv}(h)) + o(h)$, pak délka maximální cesty v daném vrcholu $U(\mathrm{Kv}(h)) = U(\mathrm{Pv}(h)) + o(h)$.

Asymptotická složitost algoritmu je $O(m + n)$, kde *m* je počet hran a *n* je počet vrcholů.

Tento algoritmus lze využít i pro zjištění acykličnosti grafu, který je rovněž v programu za tímto účelem využit. Vycházíme z věty, která je vlastností acyklických grafů:

Nechť *G* je orientovaný graf. Pak následující podmínky jsou ekvivalentní:

1. *G* je acyklický.
2. *G* nemá smyčky a každá jeho silná komponenta má jen jeden vrchol.
3. Existuje topologické uspořádání vrcholů grafu *G*.
4. Existuje topologické uspořádání hran grafu *G*.

Z toho vyplývá, že pokud nejsou po dokončení algoritmu topologického uspořádání všechny vrcholy zařazeny v posloupnosti vrcholů, pak graf není acyklický [3].

9.4.2 Algoritmus rozmístění kapacitně omezených zdrojů

V aplikaci je použit algoritmus pro rozmístění zdrojů s omezenou kapacitou pomocí metody větví a mezí. Základní popis algoritmu je popsán v předchozí kapitole. V softwarovém řešení je vytvořena i úprava tohoto algoritmu, která ovšem zpravidla nenajde všechna optimální řešení.

Pro algoritmus je použita pomocná struktura pole, která uchovává neprozkoumané listy prastromu řešení setříděné podle očekávané hodnoty ukončení projektu $f(V_i) = m + cpm$, kde m je délka cesty od příslušného listu ke kořeni a cpm je hodnota kritické cesty z podgrafu podproblému bez kapacitně omezených zdrojů. Tuto strukturu v algoritmu označíme *neprozkoumanaReseni*. Každý prvek utříděného pole se odkazuje na příslušný list do prastromu řešení. Dále je v paměti uchována proměnná $x_{nejlep\ší} = \infty$ představující nejlepší doposud nalezené řešení a posloupnost hran *Z* reprezentované například neutříděným zřetězeným seznamem.

1. krok: Nejprve vytvoříme kořen prastromu řešení. Tímto to kořenem je počáteční uzel V_0 síťového grafu. Určíme pro něj hodnotu kritické cesty bez omezení zdrojů, která je rovna hodnotě kritické cesty celého projektu. Uzel reprezentující kořen prastromu vložíme do utříděného pole *neprozkoumanaReseni*.

2. krok: Pokud je utříděné pole *neprozkoumanaReseni* prázdné, algoritmus končí.

3. krok: Vybereme první uzel V_i z utříděného pole *neprozkoumanaReseni*. Pokud $f(V_i) > f(x_{nejlep\ší})$, pak algoritmus končí, jelikož už žádný z listů nemůže nabýt lepší hodnoty, než je optimum.

4. krok: Jestliže $m = cpm$ a $f(V_i) \leq f(x_{nejlep\ší})$, pak $x_{nejlep\ší} = V_i$. Uzel V_i odstraníme z utříděného pole *neprozkoumanaReseni*. Pokračujeme krokem 3.

5. krok: Projdeme všechny činnosti předcházející podproblému vybraného vrcholu V_i směrem ke kořeni prastromu. Všechny činnosti, které

byly již vykonány celé, z grafu dočasně odstraníme a vložíme do posloupnosti *Z*, respektive pokud některé činnosti nebyly dokončeny, pak upravíme jejich ohodnocení. Následně odstraníme fiktivní činnosti, které již mají všechny bezprostředně předcházející činnosti dokončené a vložíme je do posloupnosti *Z*. Tím jsme získali podgraf odpovídající podproblému. Vybereme uzel V_j, který nemá žádnou předcházející činnost.

6. krok: Vytvoříme kombinace všech hran vycházejících z uzlu V_j, pro které je dostatek zdrojů k jejich vykonání. V této práci je řešena i úprava, že jsou vytvořena pouze ta řešení, která z množiny kombinací zatěžují zdroje nejvíce. Pro všechny kombinace opakujeme krok 6a.

6a. krok: Odstraníme činnosti odpovídající příslušné kombinaci z grafu. Vložíme do prastromu uzel V_k a určíme pro něj hodnotu $f(V_k)$. Vytvoříme hranu $[V_i, V_k]$ a vložíme ji do prastromu řešení. Odebrané činnosti kombinace vrátíme do grafu. Uzel V_k vložíme do utříděného pole *neprozkoumanaReseni.*

7. krok: Všechny hrany z posloupností *Z* vložíme zpět do grafu, případně upravíme jejich ohodnocení na původní hodnotu. Pokračujeme krokem 2.

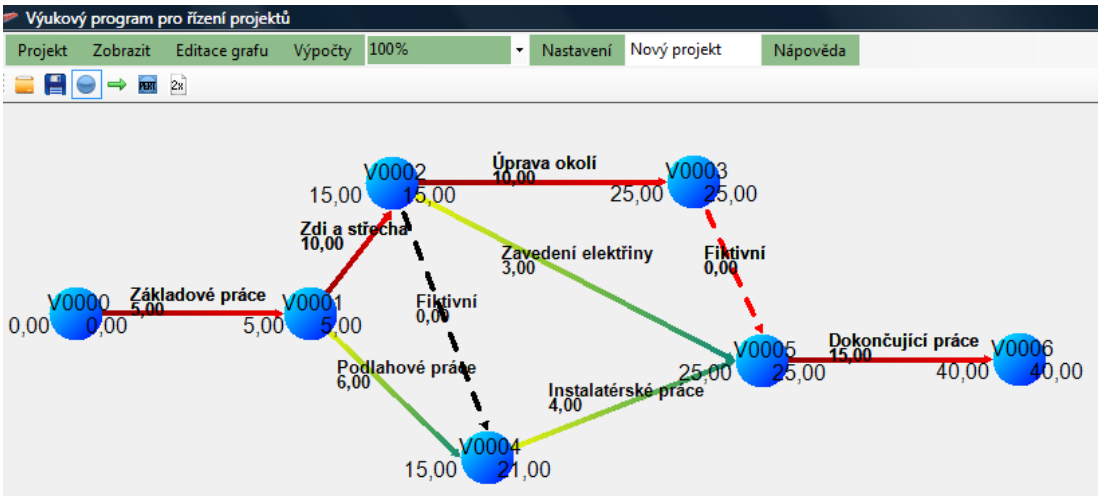
9.5 Implementace softwarového řešení

Aplikace k problematice řízení projektů je vytvořena v programovacím jazyce C# v prostředí Microsoft Visual Studio 2008. Hlavním cílem softwarového řešení je nástroj pro plánování projektů s výukovou podporou.

Vstupem aplikace je síťový graf, který může být zadán třemi způsoby. Jednou z možností je vytváření síťového grafu pomocí grafického rozhraní. Dále lze vytvářet síťový graf tabulkou, ve které se definují činnosti a činnosti jim bezprostředně přecházející. Poslední variantou vkládání grafu je možné pomocí dvou tabulek, uzlů a činností. V tomto případě se činnosti definují pomocí incidentních uzlů. Pokud uživatel zvolí zadávání pomocí tabulky, kde je síťový graf určen činnostmi s jejich bezprostředními předchůdci, pak jsou dodržena pravidla pro konstrukci síťového grafu.

Tedy problém násobných hran, či vyjadřování závislostí fiktivními činnostmi. U zbylých obou způsobů otázku správného zadání síťového grafu musí řešit uživatel. Aplikace v těchto případech zamezuje pouze zadávání násobných hran, případně smyček, které způsobují v grafu cyklus. Testování acykličnosti síťového grafu, spolu s existencí právě jednoho počátečního a jednoho koncového uzlu, jsou provedeny až při spuštění algoritmů příslušných metod. V takovém případě aplikace upozorní na nesprávně zadaná data.

Aplikace řeší plánování projektu třemi způsoby – metodou kritické cesty, metodou PERT (včetně výpočtu pravděpodobnosti vzniku kritické činnosti) a rozmístění kapacitně omezených zdrojů metodou větví a mezí. Jelikož se jedná o metody s deterministickým i stochastickým pohledem na dobu trvání činnosti, lze časové ohodnocení činností zadávat oběma způsoby.

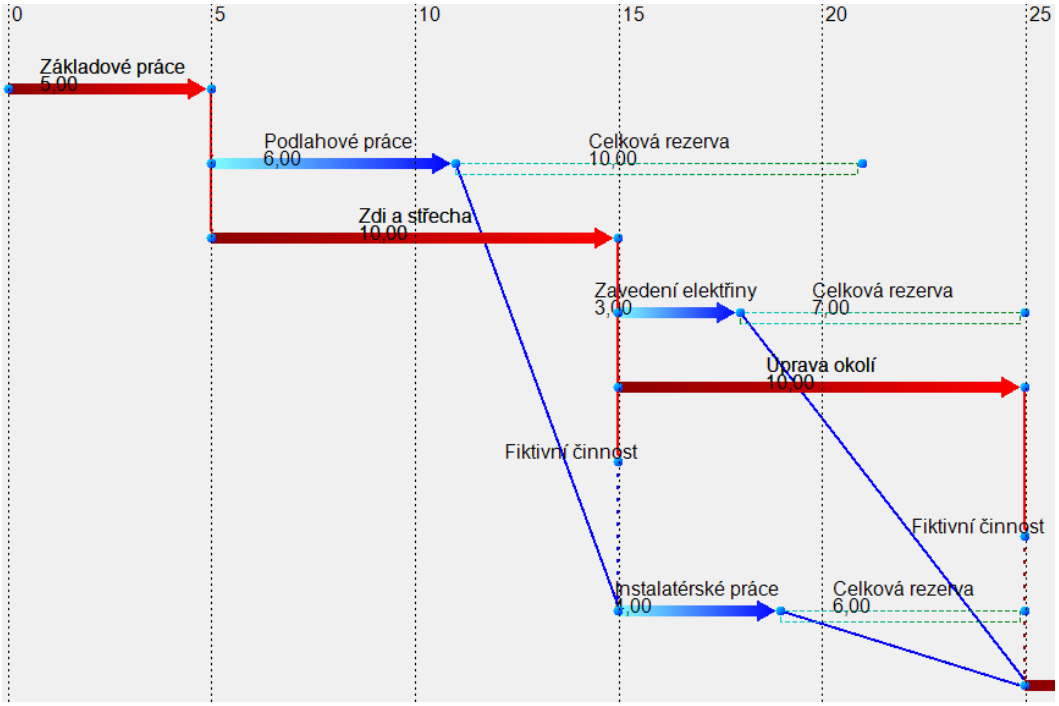


Obrázek 22 - Aplikace: Síťový graf s deterministickým ohodnocením - Zdroj: vlastní

Další vstup aplikace představují zdroje, které jsou definovány názvem, kapacitou, cenou a zda se jedná o zdroje obnovitelné, či neobnovitelné (spotřební). Se zdroji je počítáno pouze při řešení úlohy rozmístění zdrojů s omezenou kapacitou. Metody CPM a PERT se zdroji projektu nepočítají.

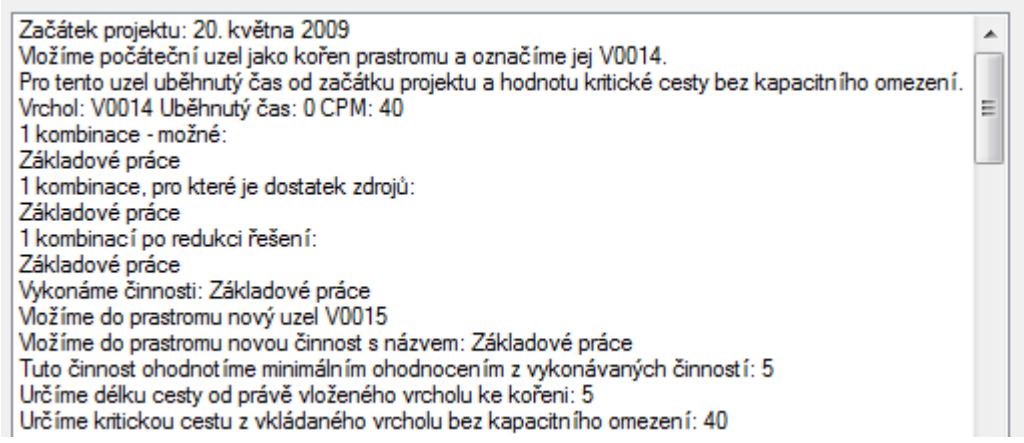
Řešení úloh pomocí metody kritické cesty a metody PERT lze graficky zobrazit i Ganttovým diagramem. Na následujícím obrázku je výřez části Ganttova diagramu odpovídající síťovému grafu na obrázku 20, kde červenou barvou je vyznačena kritická cesta, zelenou barvou celkové časové rezervy činností a modrou barvou ostatní činnosti. Výsledky úlohy rozmístění kapacitně omezených zdrojů lze zobrazit dvěma grafickými způsoby. Prvním z nich je prastrom řešení, který zobrazuje všech-

ny prozkoumané podproblémy. Zatímco druhý způsob, pomocí Ganttova diagramu, umožňuje v jednom okamžiku zobrazit pouze jediné námi zvolené řešení. Počet Ganttových diagramů pro příslušnou úlohu odpovídá počtu optimálních řešení. Z důvodů velké výpočetní náročnosti algoritmu lze v nastavení aplikace určit, zdali se mají vypočítat všechna možná řešení, nebo řešení, která jsou nalezena pomocí upraveného algoritmu. Algoritmus je podrobněji popsán v kapitole 8.4.2.



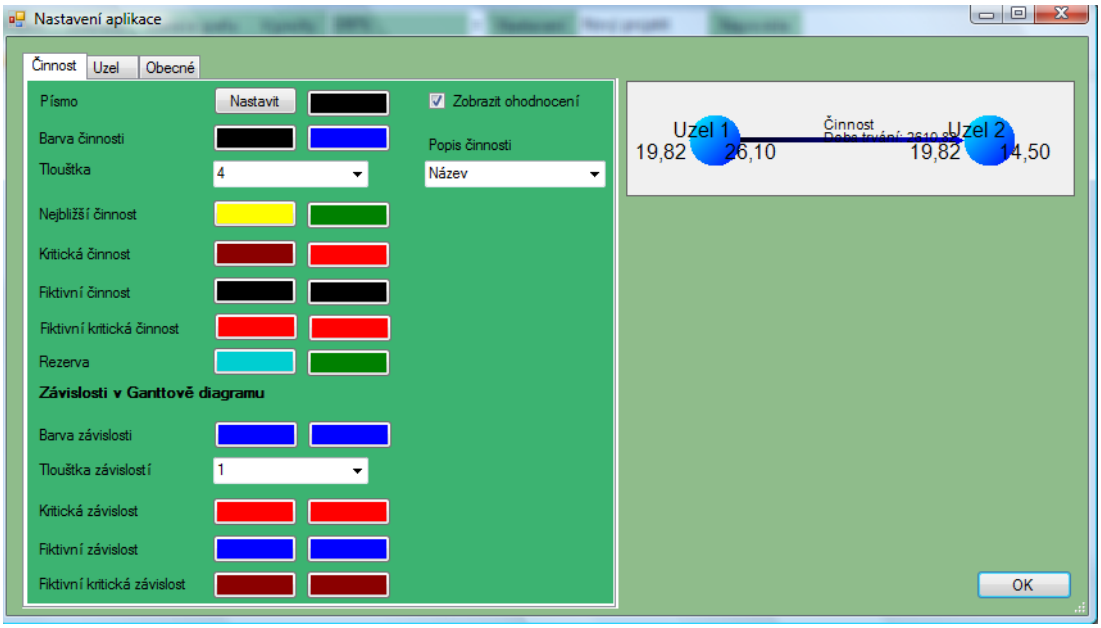
Obrázek 23 - Aplikace: Výřez Ganttova diagramu - Zdroj: vlastní

Po ukončení výpočtu libovolné metody se zobrazí statistiky projektu. Pokud je zapnut výukový mód, pak se objeví i postup algoritmu během výpočtu. Algoritmus výpočtu i výsledky úlohy lze exportovat do textového souboru.



Obrázek 24 - Aplikace: Výpis části algoritmu - Zdroj: Vlastní

Grafické zobrazení prvků síťového grafu, respektive Ganttova diagramu si uživatel může přizpůsobit v nastavení aplikace, kde lze nastavit rozsáhlé množství dalších atributů projektu. Např. začátek projektu, velikosti uzlů, fonty písem apod. Všechny projekty lze ukládat do textového, či binárního souboru. Manuál aplikace je uveden na přiloženém datovém médiu.



Obrázek 25 - Aplikace: Nastavení - Zdroj: vlastní

10 Testování efektivnosti aplikace

Jednou z testovaných úloh je výstavba městského domu, která je použita jako vzorový projektový plán. Úlohu modifikujeme ve dvou variantách a budeme sledovat různé důsledky na dobu trvání celého projektu a jeho nákladů. Následující příklad se nachází na přiloženém CD v adresáři *Projekty* pod názvem *StavbaMěstskéhoDomu.txt*.

Činnost	Uzly	Popis	Doba trvání	Předchůdce	Pracovníci
A	1-2	Základové práce	5	-	1 zedník, 4 brigádníci
B	2-3	Stavba zdí a střechy	10	A	1 zedník, 3 brigádníci
C	2-4	Podlahové práce	6	A	1 zedník, 2 brigádníci
D	3-5	Úprava terénu	10	B	5 brigádníků
E	3-6	Zavedení elektřiny	3	B	1 elektrikář, 1 brigádník
F	4-6	Instalatérské práce	4	B, C	1 instalatér, 2 brigádníci
G	6-7	Dokončovací práce	15	D, E, F	1 zedník, 3 brigádníci

Tabulka 7 - Celkový přehled projektu - Zdroj: Vlastní

V tabulce 5 jsou uvedena data projektu pro vytvoření síťového grafu. Pro úplnost projektu si zde uvedeme přehled zdrojů a jejich ocenění.

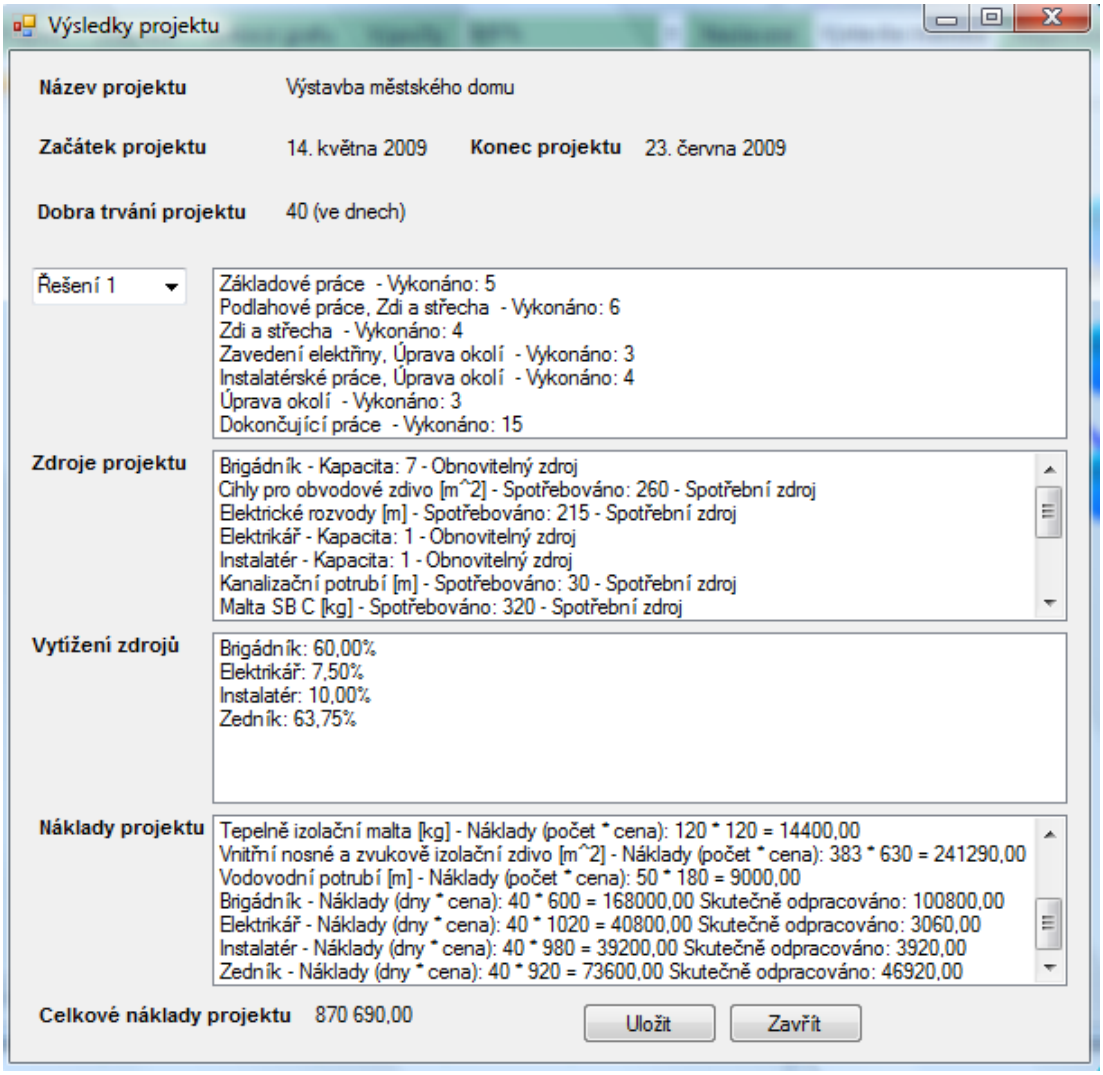
Název	Počáteční kapacita	Aktuální kapacita	Obnovitelný	Ocenění
Brigádník	7	7	ANO	600
Cihly pro obvodové zdivo [m^2]	0	0	NE	290
Elektrické rozvody [m]	0	0	NE	40
Elektrikář	1	1	ANO	1020
Instalatér	1	1	ANO	980
Kanalizační potrubí [m]	0	0	NE	240
Malta SB C [kg]	0	0	NE	115
Okna a okenní rámy [ks]	0	0	NE	10000
Střešní tašky [m^2]	0	0	NE	320
Tepelně izolační malta [kg]	0	0	NE	120
Vnitřní nosné a zvukově izolační zdivo [m^2]	0	0	NE	630
Vodovodní potrubí [m]	0	0	NE	180
Zedník	2	2	ANO	920

Obrázek 26 - Aplikace: Přehled zdrojů projektu - Zdroj: vlastní

V prvním sloupci je uveden název zdroje. Druhý a třetí sloupec zobrazují kapacity zdroje. Sloupec *Obnovitelný* určuje, zda se jedná o spotřební zdroj, či pracovní

sílu a v posledním sloupci je uvedena cena příslušného zdroje. V případě obnovitelného zdroje se jedná o cenu za 1 časovou jednotku, pro neobnovitelný zdroj je určena cena za 1 jednotku.

Začátek projektu je naplánován na 14. května 2009. Časové jednotky jsou určeny ve dnech. Při použití algoritmu, který nalezne všechna optimální řešení projektu, algoritmus prozkoumal celkem 34 podproblémů.

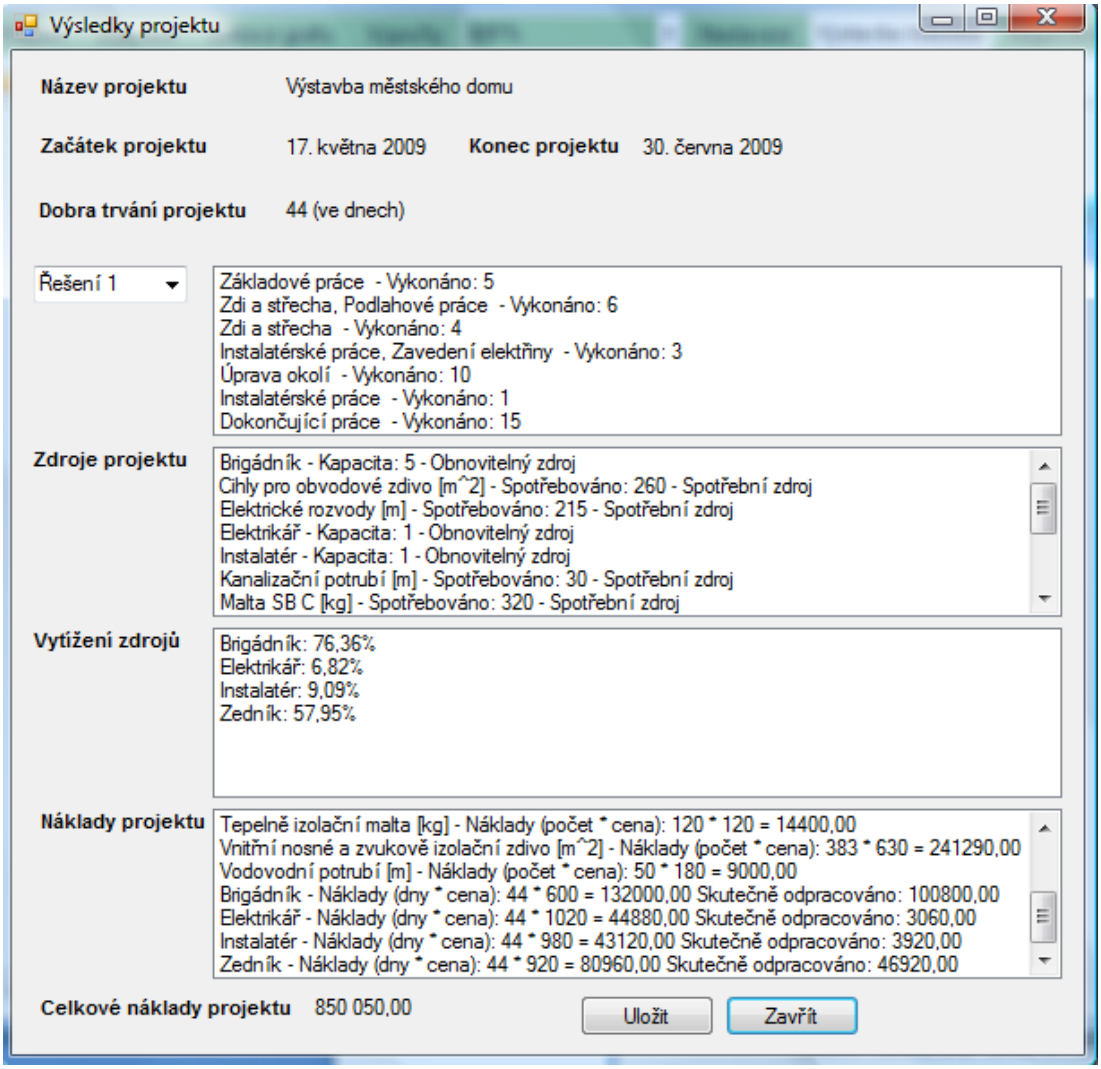


Obrázek 27 - Aplikace: Výsledky při vyhledání všech řešení úlohy - Zdroj: vlastní

Algoritmus našel celkem 2 optimální řešení. Celková doba projektu trvá 40 dní, z čehož vyplývá ukončení projektu 23. června 2009. Časové trvání projektu bez kapacitního omezení zdrojů by trvalo rovněž 40 dní. Příčinnou, že nebyla prodloužena doba trvání projektu je dostatek zdrojů pro činnosti na kritické cestě po celou dobu projektu, zatímco během této doby jsou zpracovány ostatní nekritické činnosti.

Celková cena projektu činní 870 690. Pokud spustíme algoritmus, který upřednostňuje maximální vytížení zdrojů, úloha nalezne obě optimální řešení, ale počet řešených podproblémů je redukován na pouhých 13.

Pokud snížíme počet brigádníků na 5, pak nebude již dostatečné množství pracovních zdrojů, aniž by musela být doba projektu prodloužena. V tomto případě se doba trvání projektu zvýšila na 44 dní. Počet všech možných řešení projektu vzrostl na 3 a celkem bylo řešeno 33 podproblémů. Celkové náklady klesly na 850 050. Zároveň došlo ke zvýšení vytížení zdroje *Brigádník*, ale naopak vytížení ostatních zdrojů pokleslo. Při použití druhé verze algoritmu byla nalezena pouze 2 řešení a řešeno 11 podproblémů. Tato úloha je uložena pod názvem *StavbaMěstskéhoDomu2*.



Obrázek 28 - Aplikace: Výsledky modifikované úlohy - Zdroj: vlastní

Druhý projekt byl zvolen většího rozsahu a časový pohled na činnosti je stochastický. V následující tabulce je uveden přehled činností projektu. V prvních dvou sloupcích jsou uzly (klíče) činnosti, kterými je určena. V dalších sloupcích je zkratka činnosti, název, doba trvání (v tomto případě se jedná o stochastické ohodnocení činnosti ve formátu: optimistický odhad - nejpravděpodobnější odhad - pesimistický odhad), zdroje činnosti s v závorkách uvedenou spotřebou a předchůdci činnosti. Tento projekt je uložen na přiloženém médiu pod názvem *VýrobníHala*.

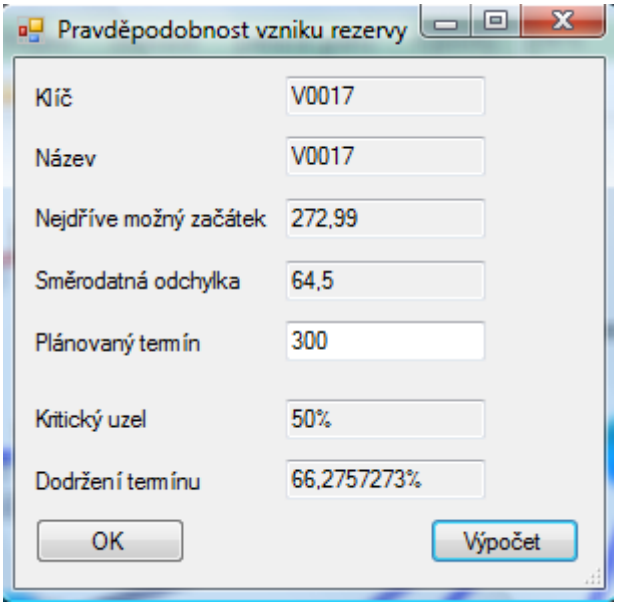
Klíč	Klíč	Zkratka	Název činnosti	Doba trvání	Zdroje	Předchůdci
V000	V001	A	Vydání stavebního povolení	14-15-17	Manažer(2)	
V001	V004	B	Hrubé terénní úpravy	10-11-15	Dělník(25),Stavební inženýr(2),Řidič(6)	A
V001	V003	C	Stavba zařízení staveniště	5-7-9	Dělník(16),Stavební inženýr(1),Řidič(5)	A
V003	V004	FH1	Fik. hr.	0-0-0	-	C
V004	V006	E	Výstavba vnějších rozvodů elektřiny	22-30-42	Dělník(8)	B,FH1
V004	V005	D	Spodní stavba výrobních hal	14-18-22	Stavební inženýr(2),Řidič(1),Dělník(14)	B,FH1
V005	V010	I	HSV pomocných objektů	32-40-49	Stavební inženýr(2),Dělník(14)	D
V005	V009	H	HSV skladů	34-38-42	Stavební inženýr(2),Dělník(21)	D
V005	V008	G	HSV správní budovy	28-33-38	Stavební inženýr(3),Dělník(26)	D
V005	V007	F	HSV výrobních hal	52-58-66	Stavební inženýr(1),Dělník(21)	D
V006	V012	K	Zhotovení podkladu železnic	8-10-13	Řidič(2),Dělník(12)	E
V006	V011	J	Zhotovení podkladu silnic	12-14-15	Řidič(4),Dělník(6)	E
V007	V013	L	PSV výrobních hal	58-65-77	Stavební inženýr(2),Dělník(12)	F
V008	V019	M	PSV správní budovy	55-60-67	Stavební inženýr(2),Dělník(14)	G
V009	V019	N	PSV skladů	57-58-63	Stavební inženýr(3),Dělník(13)	H
V010	V019	O	PSV pomocných objektů	52-58-62	Stavební inženýr(1),Dělník(8)	I
V011	V019	P	Položení povrchu silnice	8-10-12	Dělník(6),Řidič(2)	J
V012	V019	Q	Položení železniční vlečky	7-7-7	Dělník(19),Řidič(6)	K
V013	V019	R	Montáž technologických zařízení	53-62-74	Dělník(28),Stavební inženýr(2)	L
V019	V022	S	Odstraňování závad při zkušebním provozu	15-30-50	Dělník(20),Stavební inženýr(2)	M,N,O,P,Q,R
V019	V021	T	Zkušební provoz	18-18-18	Manažer(1),Stavební inženýr(2),Dělník(12)	M,N,O,P,Q,R
V021	V022	FH2	Fik. hr.	0-0-0	-	T
V022	V023	U	Předání závodu provozovatele	10-11-11	Manažer(2),Stavební inženýr(4)	S,FH2

Obrázek 29 - Aplikace: Tabulka síťového grafu - Zdroj: vlastní

Na daném projektu byl nejdříve vykonán algoritmus metody PERT, pro zjištění délky projektu bez kapacitního omezení. Celková doba trvání projektu byla přibližně 273 dní a celkem 7 činností bylo kritických. S využitím metody PERT můžeme určit, zda-li projekt stihneme dokončit během 300 dní. Na obrázku 30 jsou uvedeny výsledky výpočtu. Zkoumáme poslední uzel síťového grafu, pro který platí, že doba nejdříve možného začátku je rovna délce projektu, tedy 272,99 dní. Směrodatná odchylka projektu činní 64,5 dne. Výsledná pravděpodobnost dodržení limitu 300 dní na dokončení projektu je rovna 66,28%. Pravděpodobnost, že se uzel změní na kritický je 50%, jelikož uzel leží na kritické cestě.

Tímto jsme zjistili délku projektu bez kapacitního omezení zdrojů a nyní můžeme sledovat vliv nedostatku zdrojů, které jsou kapacitně omezeny, respektive tes-

tovat dvě verze algoritmu na rozmístění zdrojů v síti. Aplikace byla testována na počítači s operačním systémem Windows Vista, procesorem Intel(R) Core(TM)2 Duo T8300 2,4GHz a pamětí 2GB.



Obrázek 30 - Aplikace: Dodržení plánovaného termínu - Zdroj: vlastní

Nejprve byly zvoleny kapacity zdrojů uvedené na obrázku 29. Neobnovitelné zdroje byly zanedbány, jelikož nejsou předmětem zkoumání a významným způsobem neovlivňují výpočet. Zkoumaným zdrojem je *Dělník*. Činnost s největší potřebou tohoto zdroje pro vykonání využívá ke zpracování 28 jednotek. Při maximální kapacitě 52 jednotek lze velké množství činností vykonat současně.

Název	Počáteční kapacita	Aktuální kapacita	Obnovitelný	Ocenění
Dělník	52	52	ANO	900
Manažer	2	2	ANO	2800
Řidič	8	8	ANO	980
Stavební inženýr	8	8	ANO	2200

Obrázek 31 - Aplikace: Přehled zdrojů - Zdroj: vlastní

Pro upravený algoritmus, který automaticky zamítne řešení vytěžující méně zdroje projektu, jsou výsledky uvedeny v tabulce. Čas výpočtu trval řádově několik vteřin. Výsledkem bylo nalezení jediného optimálního řešení. Celková délka projektu se zvýšila o celý týden. Celkové náklady na projekt činily 21 794 421,60 Kč.

Výpočet pomocí algoritmu, který nalezne všechna řešení, časově selhal a po 3 hodinách výpočtu byl přerušen. Hlavním důvodem je, že algoritmus musí pro-

zkoumat všechna potencionální řešení. Počet potencionálních řešení značně roste díky možnosti přerušení libovolné činnosti.

Projekt: Výstavba výrobní haly			**Délka projektu**: 279,99 dní	
Začátek projektu: 13. května 2009			**Konec projektu**: 17. února 2010	
Počet nalezených optimálních řešení: 1			**Prodloužení projektu o**: 7 dní	
Náklady projektu: 21 794 421,60 Kč				
Vytížení zdrojů	Dělník	Manažer	Řidič	Stavební inženýr
	69,92%	12,50%	11,59%	50,52%

Možná varianta řešení projektu	
Název činnosti	**Vykonáno**
Vydání stavebního povolení	15,17
Stavba zařízení staveniště	7,00
Hrubé terénní úpravy	11,50
Výstavba vnějších rozvodů elektřiny, Spodní stavba výrobních hal	18,00
HSV pomocných objektů, HSV výrobních hal, Výstavba vnějších rozvodů elektřiny	12,67
Zhotovení podkladu železnic, HSV pomocných objektů, HSV výrobních hal	10,17
Zhotovení podkladu silnic, HSV skladů, HSV výrobních hal	13,83
Položení povrchu silnice, HSV skladů, HSV výrobních hal	10,00
HSV skladů, HSV výrobních hal	11,66
Položení železniční vlečky, PSV výrobních hal, HSV skladů	2,51
PSV skladů, PSV výrobních hal, HSV správní budovy	33,00
PSV skladů, PSV výrobních hal, HSV pomocných objektů	17,33
PSV pomocných objektů, PSV skladů, PSV správní budovy, PSV výrobních hal	8,34
Položení železniční vlečky, PSV správní budovy, PSV výrobních hal	4,49
PSV pomocných objektů, PSV správní budovy, PSV výrobních hal	0,16
Montáž technologických zařízení, PSV pomocných objektů, PSV správní budovy	47,34
Montáž technologických zařízení, PSV pomocných objektů	1,83
Montáž technologických zařízení	13,33
Zkušební provoz, Odstraňování závad při zkušebním provozu	18,00
Odstraňování závad při zkušebním provozu	12,83
Předání závodu provozovateli	10,83

Tabulka 8 - Výsledky projektu pro 52 přidělených dělníků - Zdroj: vlastní

V dalším případě byl upraven počet dělníků na 28, což je minimum pro vykonání libovolné činnosti projektu. Ve výsledku to znamená, že nelze již v takovém množství zpracovávat několik činností současně. Doba projektu se tím rapidně zvýšila téměř o 187 dní. I když počet zaměstnanců poklesl, celkové náklady na projekt výrazně stouply, jelikož zbylé typy pracovníků nebyly po většinu projektu využity. Z tohoto důvodu by bylo vhodné učinit případná opatření a ostatní zdroje přidělit

projektu jen v určité dny. Výpočet tohoto řešení trval řádově několik desítek minut a bylo nalezeno celkem 576 možných řešení. Příčinnou velkého množství řešení je velmi časté přerušování činností, a tím je nutné prozkoumat více variant. Mnohé z těchto činností by mohly být vykonávány souběžně, avšak v tomto případě z nedostatku zdrojů se vykonávají jednotlivě a jejich pořadí se může kombinovat s pořadím ostatních činností.

Projekt: Výstavba výrobní haly			**Délka projektu**: 459,67 dní	
Začátek projektu: 13. května 2009			**Konec projektu**: 16. srpna 2010	
Počet nalezených optimálních řešení: 576			**Prodloužení projektu o**: 186,68 dní	
Náklady projektu: 25 851 840,80 Kč				
Vytížení zdrojů	Dělník	Manažer	Řidič	Stavební inženýr
	79,09%	7,61%	7,06%	30,77%
Možná varianta řešení projektu				
Název činnosti				**Vykonáno**
Vydání stavebního povolení				15,17
Hrubé terénní úpravy				11,50
Stavba zařízení staveniště				7,00
Výstavba vnějších rozvodů elektřiny, Spodní stavba výrobních hal				18,00
HSV pomocných objektů, Výstavba vnějších rozvodů elektřiny				12,67
Zhotovení podkladu silnic, HSV skladů				13,83
Položení povrchu silnice, HSV výrobních hal				10,00
Zhotovení podkladu železnic, HSV pomocných objektů				10,17
HSV správní budovy				33,00
PSV správní budovy, HSV pomocných objektů				17,33
Položení železniční vlečky, PSV pomocných objektů				7,00
PSV pomocných objektů, PSV správní budovy				43,00
HSV výrobních hal				48,33
PSV pomocných objektů, PSV výrobních hal				7,67
HSV skladů				24,17
PSV skladů, PSV výrobních hal				58,16
PSV skladů				0,51
Montáž technologických zařízení				62,50
Zkušební provoz				18,00
Odstraňování závad při zkušebním provozu				30,83
Předání závodu provozovatele				10,83

Tabulka 9 - Výsledky projektu pro 28 přidělených dělníků - Zdroj: vlastní

10.1 Vyhodnocení efektivity algoritmů

Aplikace řeší projekty bez kapacitního omezení pomocí metod CPM a PERT v jakémkoliv rozsahu s velmi rychlou odezvou, jelikož se jedná o úlohu řešitelnou v polynomiálním čase.

Aplikace však není vhodná pro rozsáhlejší projekty s kapacitně omezenými

zdroji z důvodu NP-složitosti úlohy. Při testování aplikace na menších projektech bylo použití upraveného algoritmu efektivní. Celkový výpočet se zkrátil několikanásobně za cenu ztráty některých optimálních řešení. Velmi záleží také na kapacitě zdrojů. Při velmi nízké kapacitě roste i časová náročnost upravené verze algoritmu. Důvodem je, že algoritmus nemá k dispozici zdroje pro vykonávání více činností souběžně. Z toho vyplývá, že v tomto případě se modifikovaná verze algoritmu blíží výsledkům verze původní a musí prozkoumat velkou oblast řešení.

Celkovou náročnost úlohy podtrhuje i komplexní výpočet ve všech krocích řešení. V každé iteraci je nutné vybrat jeden z uzlů, který je potencionálně nejlepším dočasným řešením. Vytvořit podgraf z činností, které ještě nebyly vykonány a mají zpracovány všechny své předchůdce. Určit všechny možné kombinace přípustných řešení a pro jednotlivá přípustná řešení vypočítat hodnotu kritické cesty do konce projektu bez omezení zdrojů.

Pro vylepšení řešené problematiky by bylo možné učinit několik úprav, či použít některý z metaheuristických algoritmů. Jednou z možných úprav, která by ovšem pouze nepatrně snížila počet řešení, je neumožnit přerušení některých činností. Důsledek této úpravy má ovšem negativní vliv na prodloužení celkové doby projektu.

Jako lepší alternativa se jeví použití některého z metaheuristických algoritmů. Vhodnou volbou by byla metoda simulovaného žíhání, či metoda zakázaného prohledávání. Tyto algoritmy ovšem nejsou tak intuitivní a tudíž pro pochopení problematiky z pedagogického hlediska příliš vhodné. Další nevýhodou může být nenalezení optimálního výsledku, ale pouze jen jeho přibližné hodnoty. V případě využití těchto metod by byl algoritmus rozmístění použitelný i pro rozsáhlejší projekty.

Závěr

Práce se zabývá historií metod řízení projektů, teoretickými základy síťové analýzy a jejich praktickým použitím. Soustřeďuje se zejména na problematiku metody kritické cesty (CPM), metody PERT a metody větví a mezí (Branch & Bound). Pomocí metody větví a mezí je řešena problematika rozmístění zdrojů v projektu, kde jsou kapacitně omezeny zdroje.

V práci jsou popsány dva vybrané profesionální produkty a přehled několika desítek dalších softwarových nástrojů.

Praktická část práce se věnuje implementaci aplikace pro řízení projektů. Práce zahrnuje popis výsledného softwarového řešení vytvořeného v jazyku C#. Aplikace řeší plánování projektů metodou kritické cesty a metodou PERT na libovolném síťovém grafu. Úlohy s kapacitně omezenými zdroji aplikace dokáže zpracovat v přijatelném čase pro menší a středně velké projekty z důvodu NP-složitosti problému. Při použití programu ve výukovém módu aplikace vypracuje podrobný výpis postupu algoritmu.

Na dané aplikaci bylo testováno několik úloh a byla vyhodnocena efektivita použitého algoritmu z výpočetního a pedagogického hlediska. Na základě výsledků byly navrženy alternativní způsoby řešení pro problém rozmístění zdrojů v síťovém grafu.

Výsledkem práce je funkční aplikace implementující metody CPM, PERT a Branch & Bound. Aplikace zobrazuje projekty ve formě síťového grafu, Ganttova diagramu, či tabulky.

Seznam literatury

[1] VOLEK, J. *Operační výzkum I*. Pardubice: Univerzita Pardubice, leden 2002. ISBN 80–7194–410–6.

[2] KLUSOŇ, Václav. *Kritická cesta a PERT v řídící praxi*. Praha : Státní nakladatelství technické literatury, 1973.

[3] DEMEL, J. *Grafy a jejich aplikace*. Praha: Academia, 2002. ISBN 80-200-0990-6.

[4] BÍLEK, Ivo. *Aproximativní a heuristické metody řešení NP-těžkých problémů*. Vysoké technické učení v Brně, 2007. 47 s. Vedoucí diplomové práce doc. RNDr. Ing. Miloš Šeda, Ph.D. Dostupný také [online]. [cit. 2009-12-05]. Dostupný z WWW: <http://autnt.fme.vutbr.cz/szz/2007/DP_Bilek.pdf>.

[5] WEAVER, Patrick. *The Origins of Modern Project Managment* [online]. 2007 [cit. 2009-05-12]. Dostupný z WWW: <http://www.mosaicprojects.com.au/PDF_Papers/P050_Origins_of_Modern_PM.pdf>.

[6] WEAVER, Patrick. *A Brief History of Sheduling : Back to the Future* [online]. 2006 [cit. 2009-05-13]. Dostupný z WWW: <http://www.pmforum.org/library/papers/2006/A_Brief_History_of_Scheduling.pdf>.

[7] TOMAŠTÍK, Marek. *Personální management firmy Baťa, a.s. Zlín do roku 1939*. Zlín, 2008. Univerzita Tomáše Bati ve Zlíně. Vedoucí dizertační práce Prof. Ing. František Trnka, Csc. Dostupný také [online]. [cit. 2009-13-05]. Dostupný z WWW: <https://www.stag.utb.cz/apps/stag/dipfile/index.php?download_this_unauthorized=10726>.

[8] *Branch and Bound* [online], Wikipedia, 24.2.2009 [cit. 2009-05-13]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Branch_and_bound>.

[9] *Simulated Annealing* [online]. Wikipedia, [cit. 2009-05-13]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Simulated_annealing>.

[10] *Tabu Search* [online]. 1997 [cit. 2009-04-03]. Dostupný z WWW: <http://www.cs.sandia.gov/opt/survey/ts.html>.

[11] NEČAS, Jiří. *Grafy a jejich použití*. Praha : Státní nakladatelství technické literatury, 1978.

[12] MAKEOWER, M. S, WILLIAMSON, E. *Teach Yourself Operational Research, Problems, Technique and Excercises*. Hodder Arnold H&S, 1985.

[13] SEDLÁČEK, Jiří. *Kombinatorika v teorii a praxi*. Praha : Československá akademie věd, 1964.

[14] LEE, Sang M., OLSON, David L. *Introduction to Management Science*. Mason : Thomson, 2006. 690 s. ISBN 0-324-41599-0.

[15] CLAUSEN, Jens. *Branch and Bound Algorithms - Principles and Examples*. Kodaň : University of Copenhagen, 1999. Dostupný z WWW: <http://www1.imada.sdu.dk/Courses/DM85/TSPtext.pdf>.

[16] ŠTEFKA, David. *Alternativy k evolučním optimalizačním algoritmům*. České vysoké učení technické v Praze, 2005. 100 s. Vedoucí diplomové práce Ing. RNDr. Martin Holeňa, CSc. Dostupný také [online]. [cit. 2009-13-05]. Dostupný z WWW: <http://www.insophy.cz/doc/stefka_optimalizace.pdf>.

[17] RUDOVÁ, Hana. *Lokální prohledávání* [online]. 2009 [cit. 2009-05-07]. Dostupný z WWW: <http://www.fi.muni.cz/~hanka/rozvrhovani/prusvitky/druha_bw.pdf>.

[18] GLOVER, Fred, LAGUNA, Manuel. *Tabu Search*. Colorado : Hard-bound, 1997. 408 s. ISBN 0-7923-8187-4.

[19] KALÁTOVÁ, Eva, DOBIÁŠ, Jaroslav. *Biologická podstata evolučních algoritmů* [online]. 2000 [cit. 2009-05-13]. Dostupný z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/gen_alg2/index.htm>.

[20] *List of project management software* [online]. 2003 [cit. 2009-05-13]. Dostupný z WWW: <http://en.wikipedia.org/wiki/List_of_project_management_software>.

ÚDAJE PRO KNIHOVNICKOU DATABÁZI

NÁZEV PRÁCE	Výukový program pro řízení rozsáhlých projektů
AUTOR PRÁCE	Šváha Martin
OBOR	Informační technologie
ROK OBHAJOBY	2009
VEDOUCÍ PRÁCE	Doc. Ing. Josef Volek, Csc.
ANOTACE	Výukový program pro řízení projektů metodami síťové analýzy
KLÍČOVÁ SLOVA	CPM, PERT, metoda větví a mezí, složitost úloh