WP 1.F Národní monitoring oběhového hospodářství

1.F.2.3

Dokumentace a manuál nástroje

Nástroj pro rekonstrukci vybraných toků odpadů na území ČR

REkonstrukce a **VE**rifikace **DAT** z IS**O**H (REVEDATO)

Kolektiv autorů:

Ing. Jaroslav Pluskal
Ing. Radovan Šomplák, Ph.D.
doc. Ing. Martin Pavlas, Ph.D.

Řešitelské pracoviště:

Ústav procesního inženýrství, VUT v Brně

Květen 2023

T A
Č R

Projekt SS02030008 **Centrum environmentálního výzkumu: Odpadové a oběhové hospodářství a environmentální bezpečnost (CEVOOH)** je financován se státní podporou Technologické agentury ČR a Ministerstva životního prostředí ČR v rámci **Programu Prostředí pro život**.

www.tacr.cz www.mzp.cz

Obsah

1 Úvod

Následující dokument poskytuje ucelený uživatelský návod pro práci se softwarem REVEDATO, jenž slouží k rekonstrukci a verifikaci dat z odpadového hospodářství. Cílem nástroje je poskytnou konzistentní databázi bez hmotnostních nesouladů, která následně může být předmětem statistického vyhodnocení a analýz pro stanovení správného směřování vývoje s ohledem na platnou legislativu a stanovené ambiciózní cíle. Nástroj umožňuje provádět výpočty na úrovní základní územní jednotky (ZÚJ) bez rozlišení typu subjektu, přičemž uživatelské rozhraní je implementováno do prostředí MS Excel s využitím výpočtového externího modulu v jazyce Python. Jádrem výpočtového modulu je optimalizační proces, který probíhá v souladu s přístupy k vyrovnání dat, které zajistí hmotnostní bilance na všech úrovních systému s minimálními odchylkami od reportovaných hodnot. Software se skládá z následujících souborů:

- REVEDATO.xlsm
- VyrovnaniDat.exe

Tyto soubory musí být ve stejném adresáři a není povoleno je přejmenovávat. Veškerá obsluha nástroje je řízena v rámci souboru „REVEDATO.xlsm“ v listu „MENU“. Vývoj nástroje probíhal ve verzi Microsoft Office Proffesional Plus 2019 s využitím jazyka Visual Basic, přičemž při spuštění souboru je nutné povolit makra. V případě jiných verzí MS Office není zaručena kompatibilita a funkčnost všech implementovaných prvků.

2 Definice pojmů

V rámci manuálu a dokumentace jsou využívány pojmy navázané na legislativu, stejně jako nově zavedené výrazy pro snadnější a výstižné odkazování na specifické názvy.

- **ZÚJ** – Zkratka pro označení „Základní územní jednotka“, která definuje nejmenší administrativní členění ČR, tj. z pohledu výkonu veřejné správy se již dále nečlení.
- **ORP** – Zkratka pro označení „Obec s rozšířenou působností“, která definuje samosprávu nadřazenou členění ZÚJ. Jedná se tedy o mezičlánek mezi krajskými úřady a ostatními obecními úřady.
- **ISOH** – Zkratka pro „Informační systém odpadového hospodářství“, jenž představuje ucelený systém pro potřeby řízení a kontroly odpadového hospodářství. Obsahuje reportovaná data o nakládání s odpady od všech subjektů.
- **PDISOH** – Představuje „Pracovní databázi ISOH“, která oproti reportovaným záznamům od subjektů obsahuje navíc dopočítané hodnoty, přepočty a případné korekce či rekonstrukce toků odpadů.
- **Evident** – pojem označuje ZÚJ, které daný záznam o nakládání s odpadem do databáze ISOH reportuje.

- **Partner** – pojem označuje ZÚJ, které evident uvedl v rámci reportu o nakládání s odpadem a na které má nakládání návaznost. Hlavní význam je v případě předání a převzetí odpadu (viz níže), aby bylo možné správně identifikovat tok odpadu.
- **Katalogové číslo odpadu** – Dle platné legislativy jsou odpady děleny do několika skupin, označeny 6místným číselným kódem. Podrobnosti a seznam jednotlivých katalogových čísel lze dohledat ve vyhlášce Ministerstva životního prostředí (MZP, 2021).
- **Indikátor odpadového hospodářství** – Soustava indikátorů popisuje stav životního prostředí. Těmito indikátory jsou popisovány systémové analýzy vztahů mezi životním prostředím a lidskou činností. Hlavním cílem indikátorů je poskytování informací o stavu a vývoji v dané oblasti. Podrobnosti a jejich matematické vyjádření je možné dohledat v metodice vypracované Ministerstvem životního prostředí (MZP, 2022).
- **Kód nakládání** – jedná se o kódové označení příslušné operace či manipulace s odpadem, jehož množství je v rámci výkazu reportováno. Kódy lze rozdělit na **kladnou část** (odpad vzniká, tj. produkce, nebo se kumuluje) a **zápornou část** (odpad zaniká, tj. zpracování, nebo je předáván). Dále je možné je rozdělit na kódy **s partnerem** a **bez partnera**. Kódy s partnerem mají návaznost pouze na samotného evidenta a v tomto případě je partner vždy roven evidentovi. Naopak kódy s partnerem označují nakládání, které má návaznost i na jiný subjekt v systému. Kódy nakládání dále začínají písmenem (obecně označeného jako **X**), které uvádí původce odpadu. Konkrétní možnosti jsou následující:
  - **A** – odpad, který byl vyprodukován stejným subjektem, který dané nakládání vykazuje.
  - **B** – odpad, který byl vyprodukován jiným subjektem, než který dané nakládání vykazuje.
  - **C** – odpad, který zůstal vykázán z předchozího kalendářního roku stejným subjektem, který dané nakládání vykazuje.
- **Produkce** – Označení zahrnuje veškeré nakládání s odpadem, které představuje kladnou část bez partnera. Jako příklad lze uvést kódy A00, C00, BN30, BN40.
- **Zpracování** – Označení zahrnuje veškeré nakládání s odpadem, které představuje zápornou část bez partnera. Jako příklad lze uvést kódy XR1, XR5, XN5.
- **Předání** – Označení zahrnuje veškeré nakládání s odpadem, které představuje zápornou část s partnerem. Konkrétní kódy jsou následující: XN2, XN3, XN8, XN10.
- **Převzetí** – Označení zahrnuje veškeré nakládání s odpadem, které představuje kladnou část s partnerem. Jako příklad lze uvést jediný kód B00.

3 Struktura vstupních dat

Vstupní data pro nástroj musí být nachystána ve struktuře, kde jednotlivé řádky představují dílčí záznamy s odpovídajícími informacemi se 23 sloupci. První parametr záznamu se týká roku, za který bylo dané nakládání vykázáno. Další informaci představuje definice evidenta a partnera záznamu. Postupně je nutné zahrnout informaci o ZÚJ, ORP, kraji a číselného typu subjektu (firma, obec atd.), přičemž musí být uvedeno jak kódové označení, tak odpovídající název. Záznam následně musí obsahovat informaci o typu odpadu, označení způsobu nakládání dle kódů definovaných legislativou a odpovídající množství přidělené dle kategorie záznamu na kladnou či zápornou část. **Data je nutné doplnit o hlavičku v přesně daném pořadí, která je vizualizována na Obr. 1.** V tomto manuálu jsou z důvodu zachování čitelnosti sloupce uspořádány do tří řádků, avšak v nástroji jsou uvedeny na jednom řádku.

Rok	Evident - kód ZÚJ	Evident - název ZÚJ	Evident - kód ORP	Evident - název ORP	Evident - kód kraje	Evident - název kraje	Evident - Typ subjektu
	Partner - kód ZÚJ	Partner - název ZÚJ	Partner - kód ORP	Partner - název ORP	Partner - kód kraje	Partner - název kraje	Partner - Typ subjektu
Katalogové číslo odpadu	Název odpadu	Kategorie odpadu katalogová	Kategorie odpadu dle původce (ohlášená)	Kód nakládání	Název kódu nakládání	Množství odpadu + (t)	Množství odpadu - (t)

Obrázek 1: Načítání dat z externích souborů.

Sloupce zvýrazněné žlutou barvou jsou pro nástroj klíčové a bez jejich zadání není možné realizovat analýzy či rekonstrukce. Ostatní informace o záznamech představují doplňující informaci pro uživatele, případně je možné jejich využití v budoucnu při dalších rozšířeních. Doplňující údaje tedy není nutné uvádět, avšak je striktně nutné zachovat navrženou strukturu, tj. je možné dané sloupce ponechat prázdné. Ukázková sada dat je přiložena k souborům softwaru pod označením „Transakce.xlsx“.

4 Uživatelský manuál k software

Následující kapitola je věnována podrobnému popisu jednotlivých listů v softwaru implementovaného v prostředí MS Excel. Detailně je zde vysvětlen postup včetně vysvětlivek k dílčím volitelným funkcionalitám, které jsou uživateli během analýz k dispozici.

4.1 List „Úvod“

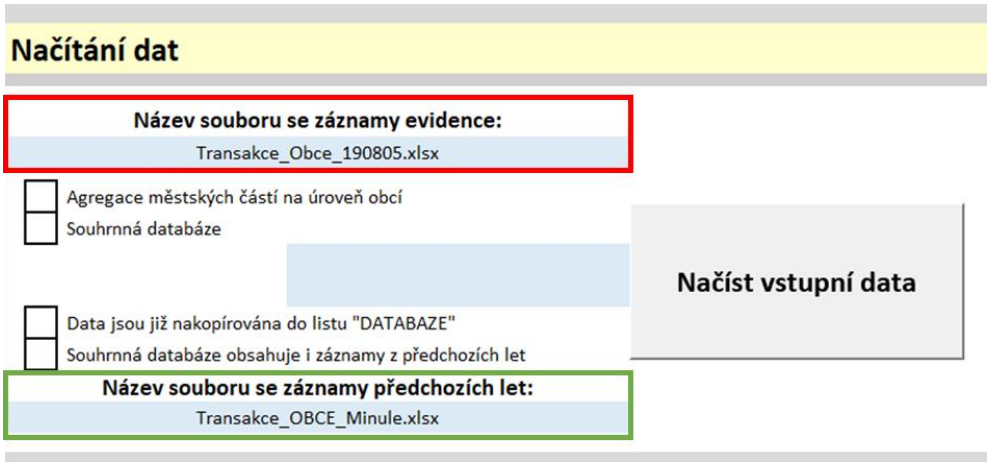
List obsahuje základní informace o řešeném projektu, kolektivu autorů a softwaru včetně jeho aktuální verze. Součástí jsou také dílčí popisky, vysvětlivky k používaným termínům a stručným návodem, jak software používat.

4.2 List „MENU“

Tento list představuje řídící jádro celého softwaru a uživatel nástroje pomocí tohoto rozhraní ovládá veškeré funkcionality nástroje. Současně jsou k dispozici souhrnná vyhodnocení dílčích analýz. V celém listu je povoleno uživateli modifikovat pouze modře podbarvená pole. Ostatní hodnoty jsou doplňovány za běhu programu automaticky. Funckionality jsou rozděleny do několika ucelených bloků, které jsou spouštěny separátně v posloupnosti shora dolů.

4.2.1 Sekce „Načítání dat“

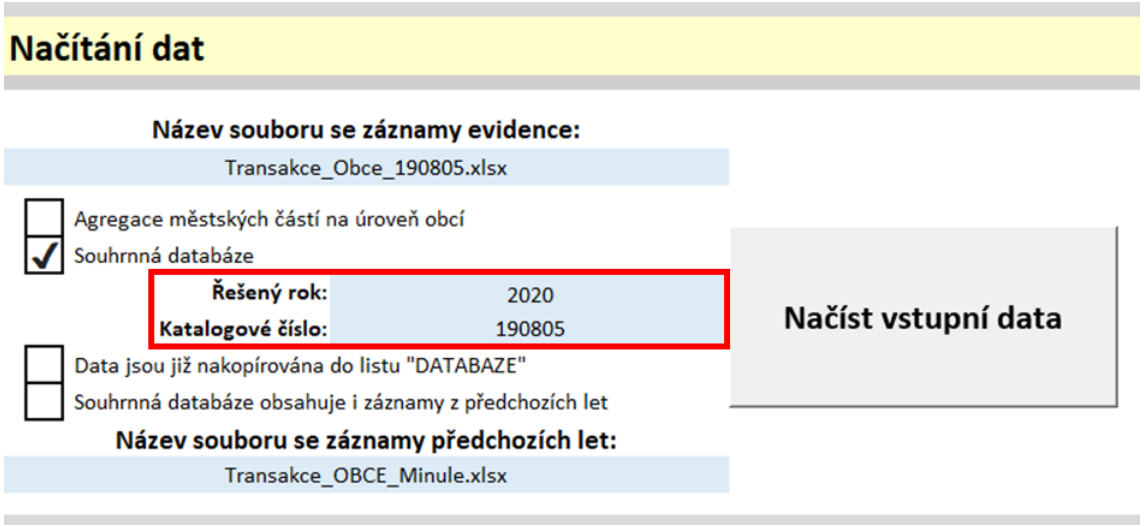
První sekce je věnována načítání dat pro následné analýzy, vyhodnocení a rekonstrukce. Výchozí nastavení nástroje je pro import dat z externího souboru, jenž musí splňovat podmínky a strukturu uvedenou v kapitole 3. Uživatel musí tento soubor umístit do stejného adresáře jako tento software, přičemž je nutné uvést jeho název (viz červené pole na Obr. 2).



Obrázek 2: Sekce „Načítání dat“ – Načítání dat z externích souborů.

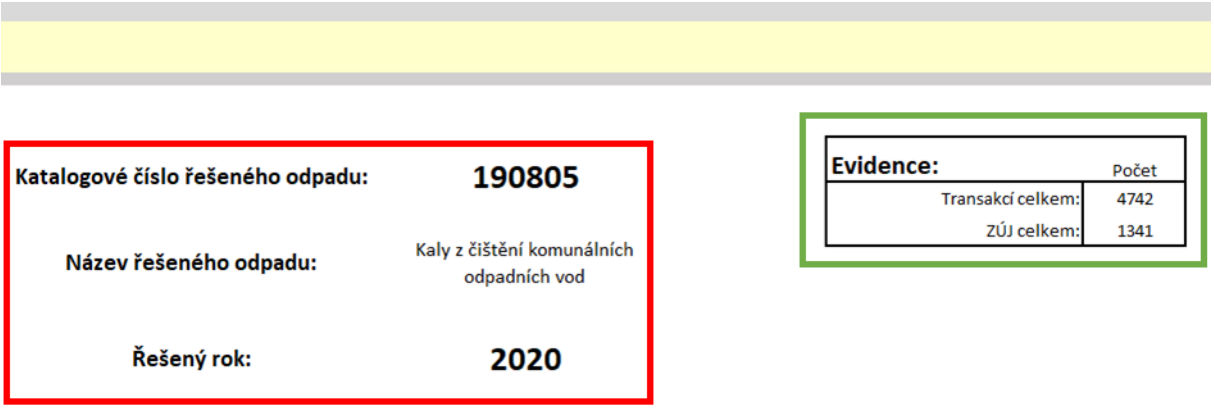
Načítání dat obsahuje několik dílčích funkcionalit, které si uživatel může navolit dle vlastních preferencí. V některých případech je možné aktivovat pouze vybrané kombinace, které jsou z podstaty věci možné (např. současné načítání ze souboru nebo ruční vložení). Výčet bodů včetně popisu je následující:

- Funckionalita „Agregace městských částí“ slouží ke spojení velkých měst, jež jsou rozděleny na další administrativní celky. Z pohledu oprav chyb, před samotným vyrovnáním dat, může být tímto krokem dosaženo lepšího výsledku rekonstrukce, avšak je částečně ztracena informace o dílčích ZÚJ.
- Alternativní zadávání je pomocí externího souboru obsahující souhrnnou databázi. Souhrnnou databází je myšlena datová sada, která obsahuje vícero typů odpadů, stejně tak jako evidenci z různých roků. Po zakliknutí příslušného pole je možné pomocí nově zpřístupněných buněk (viz červené pole na Obr. 3) zadat požadovaný kód odpadu a rok.
- Vstupní data je možné také manuálně vložit do listu „DATABAZE“. Příslušné pokyny pro uživatele jsou uvedeny v kapitole 4.3. Po vložení dat je nutné potvrdit tuto volbu zakliknutím příslušného bodu. Zde je nutné zdůraznit, že po realizaci dalších úprav samotným softwarem může dojít ke změně ručně zadaných dat (např. funckionalita agregace městských částí). Pro restart analýz a vyhodnocení je nutné data ručně nakopírovat znovu.
- Jelikož souhrnná databáze může obsahovat různé části evidence včetně dat z různých kalendářních roků, je umožněno uživateli importovat z tohoto souboru také data týkající se minulých let pro volitelné analýzy vývoje v bloku „Zpracování dat“.



Obrázek 3: Sekce „Načítání dat“ – Využití souhrnného externího souboru.

Po zadání požadovaného formátu importu dat je po stisknutí tlačítka „Načíst vstupní data“ provedena příslušná operace. Celý proces může zabrat několik desítek vteřin až jednotky minut dle velikost načítaných dat. Vliv na časovou náročnost má také velikost souhrnné databáze, zde je tedy uživateli doporučeno, aby v souboru byla obsažena pouze data týkající se nejbližších výpočtů. **Během procesu může MS Excel zdánlivě vypadat, že přestal pracovat. Je nutné patřičnou dobu počkat až do objevení informativního okna o dokončení procesu.** Následně jsou pro uživatele k dispozici informativní tabulky (viz Obr. 4).



Obrázek 4: Informační tabulky po načtení vstupních dat.

V červeném poli je k vidění rekapitulace zadaných parametrů pro požadovaná data. Zároveň je zde pro kontrolu také uveden celý název katalogového čísla. Zelené pole obsahuje informaci o počtu importovaných záznamů, stejně tak o počtu ZÚJ, které v importované databázi figurují.

4.2.2 Sekce „Zpracování dat“

Ke zpracování dat je možné přistoupit pouze, pokud byla korektně importovaná data z databáze PDISOH, v opačném případě není uživateli umožněno pokračovat. Zpracování dat identifikuje jednotlivé záznamy, přiřadí je do správných kategorií (produkce, zpracování, předání, převzetí) a provede nezbytné analýzy pro výpis souhrnných statistik a potřeby následné rekonstrukce. Uživatel má však k dispozici několik volitelných funkcionalit (viz Obr. 5), které mohou poskytnout detailnější informace o chybách v databázi či indikátorech odpadového hospodářství.

Obrázek 5: Sekce „Zpracování dat“ – Volitelné funkcionality během zpracování dat a následných analýz.

Následující bodový výpis popisuje jednotlivé funkcionality, které je možné zvolit nezávisle. Jedinou výjimkou je historický vývoj produkce a nakládání, jelikož je tato funkce implementována do stejného výstupu jako Nakládání s odpady dle původu.

- Kontrola evidence předúpravy odpadu umožňuje získat hmotnostní bilanci u jednotlivých ZÚJ týkající se předúpravy odpadu a následného vykázání sekundární produkce pod kódem BN40. Nástroj z pohledu předúprav bere v úvahu kódy nakládání XD8, XD9, XD13, XD14, XR12. V případě, že množství úpravy odpadu není rovno vykazovanému množství BN40, nástroj tuto skutečnost uvede jako chybu včetně výpisu procentuálního úbytku (či nárůstu) odpadu. Zde je však nutné upozornit, že část upravovaného odpadu může být následně přesunuto do jiného katalogového čísla, proto nemusí být tento nesoulad chybou. **Pro zobrazení výstupů kontroly je nutné povolit funkcionalitu „Výpis chyb“.**
- Kontrola uskladnění odpadu a následného vyskladnění v následném roce umožňuje uživateli získat informace o nesouladech v meziročním porovnání. **Pro tuto kontrolu je nutné mít k dispozici data i z minulých let a uvést je v předchozí sekci „Načítání dat“.** Předmětem analýz jsou kódy nakládání XN5, XD15 a XR13 (uskladnění odpadu) z evidence týkající se předchozího roku, než je aktuálně řešen. Pokud množství uskladněného množství odpadu neodpovídá množství odpadu vykázaném pod kódem C00 v řešeném roce, nástroj příslušné ZÚJ uvede jako chybné v příslušném výpise. **Pro zobrazení výstupů kontroly je nutné povolit funkcionalitu „Výpis chyb“.**
- Souhrnná analýza za každé ZÚJ poskytne uživateli ucelený přehled o produkci a nakládání s vybraným odpadem v řešeném roce, jenž budou k dispozici v listu „ISOH – statistika“. Součástí je také vyhodnocení indikátorů jak za celou ČR, tak i v detailu jednotlivých krajů včetně informací o přesunu odpadu mezi nimi. Podrobný popis výstupu je uveden v popisu příslušného listu v kapitole 4.5.

- Nakládání s odpady dle původce představuje stejné vyhodnocení jako předchozí bod, ovšem jsou zde zohledněny jen vybrané kódy nakládání dle počátečního písmena. Příslušné výstupy vyhodnocení jsou následně k dispozici na žlutě podbarvených listech.
- Předchozí analýzu nakládání s odpadem je možné doplnit i o meziroční srovnání a získat představu o historickém vývoji. Z důvodu náročnosti celého procesu není možné zohlednit více jak dvě předešlá období. **Tyto údaje musí být v příslušném souboru dle nastavení v sekci „Načítání dat".**
- Funkcionalita „Výpis chyb" umožňuje uživateli vypsat chybové hlášení týkající se kontrol základních hmotnostních bilancí u jednotlivých ZÚJ včetně předání a převzetí mezi nimi. Výsledky jsou následně k dispozici na listu „Výpis chyb", kde jsou také uvedeny výstupy volitelných kontrol (předúpravy odpadu a evidence skladu).

Po volbě požadovaných funkcionalit je možné spustit proces stisknutím tlačítka „Statistická analýza", který může trvat několik desítek vteřin až jednotky minut dle velikost analyzovaných dat. **Během procesu může MS Excel zdánlivě vypadat, že přestal pracovat. Je nutné patřičnou dobu počkat až do objevení informativního okna o dokončení procesu.** Následně jsou krom detailních výstupů na zvláštních listech pro uživatele k dispozici informativní tabulky (viz Obr. 6).

Základní statistika:	Množství [t]
Produkce:	954169.653
Zpracování:	971128.344
Převzetí:	1065059.051
Předání:	1048100.36

Základní chyby:	Množství [t]
Kumulativní chyba bilance ZÚJ:	0
Kumulativní chyba předání:	339281.895
Rozdíl bilance celé ČR:	-16958.691
Rozdíl v předání – převzetí:	-16958.691

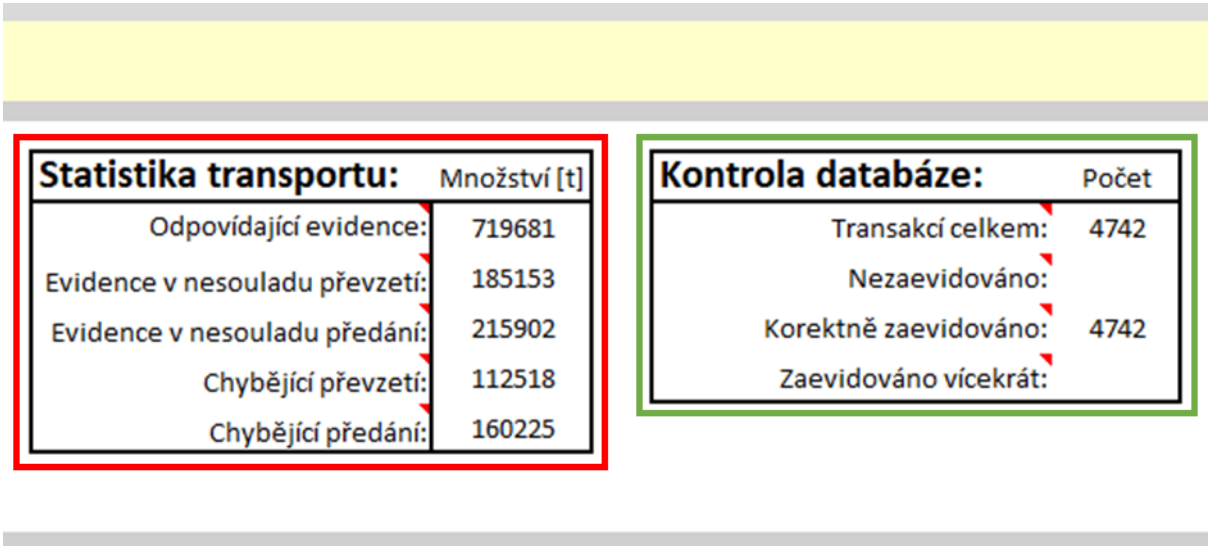
Obrázek 6: Sekce „Zpracování dat" – Základní vyhodnocení po zpracování dat.

Červené pole obsahuje informace tykající se množství odpadu v jednotlivých kategoriích. Jedná se o produkci odpadu (kumulace odpadu, tj. veškeré kladné kódy nakládání krom převzetí odpadu), zpracování odpadu (veškeré záporné kódy nakládání krom předání

odpadu), převzetí odpadu (kód nakládání B00) a předání odpadu (kódy nakládání XN2, XN3, XN8, XN10). Zelené pole představuje vyhodnocení chybovosti analyzované evidence, jehož dílčí body jsou následující:

- Kumulativní chyba bilance představuje součet hmotnostních nesouladů u jednotlivých ZÚJ bez ohledu na výsledné znaménko bilance (od kladných kódů nakládání jsou odečítány záporné kódy nakládání). Cílem tohoto indikátoru je poskytnout informaci o chybovosti bilancí v uzlech a v celém systému, jenž může být získaná například při porovnání s celkovou produkcí odpadu.
- Kumulativní chyba předání je vypočítána jako součet hmotnostních nesouladů u předání a převzetí odpadu mezi jednotlivými ZÚJ. Stejně jako u přechozího bodu není rozlišeno znaménko výsledné bilance, jelikož ukazatel má poskytnout informaci o celkové chybovosti transportu odpadu, která může být získána při porovnání s hodnotou celkového předání, převzetí či jejich průměru.
- Rozdíl bilance celé ČR uvádí celkový nesoulad mezi produkcí a zpracováním, přičemž kladné číslo značí ztrátu odpadu (tj. existuje odpad, který nebyl zpracován a byl v některé části řetězce vypuštěn z evidence) a záporné číslo značí nárůst odpadu (tj. byl zpracován odpad, který nemá jasně definovaný původ). Rozdíl oproti kumulativní chybě bilance je způsoben vyrušením nesouladů při zohlednění znaménka, zároveň tuto hodnotu ovlivňují nesoulady v předání a převzetí.
- Rozdíl v předání a převzetí, stejně jako u přechozího bodu, reflektuje hmotnostní rozdíl mezi těmito skupinami, přičemž se bere v úvahu znaménko dílčích záznamů.

Uživatel má v přehledu také k dispozici další dvě tabulky (viz Obr. 7) navázané na vyhodnocení chybovosti předání a převzetí a souhrnnou kontrolu databáze, zda-li byly všechny importované záznamy korektně identifikovány dle kódu nakládání.



Statistika transportu:	Množství [t]
Odpovídající evidence:	719681
Evidence v nesouladu převzetí:	185153
Evidence v nesouladu předání:	215902
Chybějící převzetí:	112518
Chybějící předání:	160225

Kontrola databáze:	Počet
Transakcí celkem:	4742
Nezaevidováno:	
Korektně zaevidováno:	4742
Zaevidováno vícekrát:	

Obrázek 7: Sekce „Zpracování dat“ – Statistika transportu a kontrola databáze.

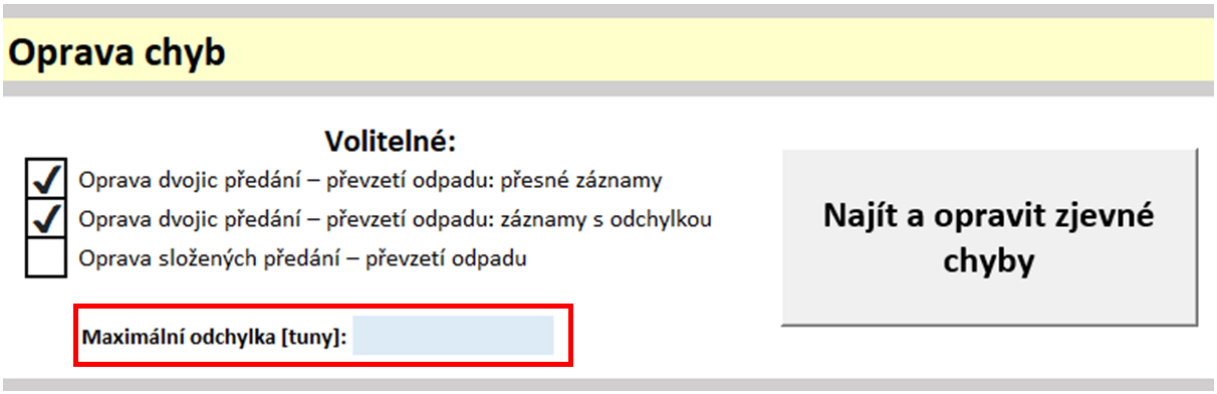
Červené pole uvádí základní údaje o transportu odpadu (předání a převzetí) na řešeném území. Následující body popisují význam uvedených hodnot.

- Jedná o množství odpadu v rámci vykazovaného předání a převzetí, které v evidenci splňuje hmotnostní bilance a správné uvedení partnerů v evidenci.
- Evidence v nesouladu představuje předání a převzetí v evidenci, kde existují příslušné výkazy ke spárování, avšak hmotnostní bilance není rovna nule. Tj. evidované množství odpadu při předání není rovno evidovanému množství při převzetí. Kategorie je rozdělena do dvou částí na předání a převzetí, aby bylo možné posoudit celkový nesoulad mezi transakcemi. V uvedeném případě bylo předáno přibližně o 30 kt více odpadu, než bylo následně převzato (viz Obr. 7).
- Následující dvě skupiny jsou vztaženy k části databáze výkazů předání a převzetí, u které nebylo možné dohledat odpovídající záznam do páru. Tj. v případě předání neexistuje záznam o převzetí („Chybějící převzetí“) a v případě převzetí není dohledáno odpovídající předání („Chybějící předání“).

Kontrola databáze v zeleném poli uvádí, kolik je v evidenci celkem záznamů a zároveň poskytuje informaci, jestli byly všechny korektně identifikovány a zařazeny. Pokud nebylo možné záznam identifikovat, či byl z nějakého důvodu zařazen do nějaké kategorie vícekrát, uživatel je v rámci tabulky (viz zelené pole na Obr. 7) o této skutečnosti informován. Není doporučeno uživateli pokračovat v dalších krocích, pokud nejsou všechny záznamy korektně zaevidovány.

4.2.3 Sekce „Oprava chyb“

Tato sekce je věnována opravám chyb v předání a převzetí před samotným bilančním algoritmem. Realizace těchto oprav není vyžadována, avšak z pohledu rekonstrukce je doporučeno provést alespoň dílčí opravy, které mohou snížit počet stupňů volnosti v řešeném systému. Následující Obr. 8 vizualizuje příslušnou zadávací část v softwaru.



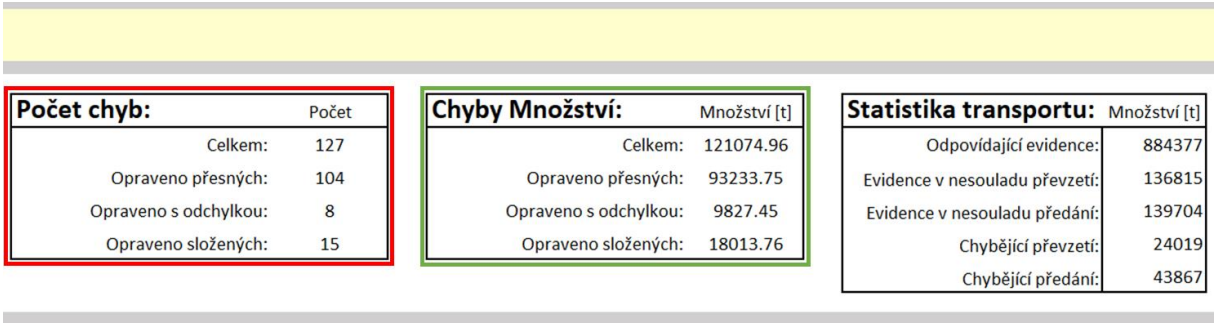
Obrázek 8: Sekce „Oprava chyb“ – Volitelné funkcionality.

Opravy probíhají na základě porovnávání hmotnostních nesouladů mezi jednotlivými záznamy a následné úpravy vykázaných partnerů. Často jsou k vidění chyby, které plynou ze špatného vykázání předání na sídlo firmy v souladu s fakturací, avšak ve skutečnosti byl odpad dopraven pouze na pobočku, která tuto skutečnost eviduje. Podrobnější informace ohledně opravy podobných chyb a názorných příkladů lze dohledat v dokumentu M1F2 k V1.F.2.1 (CEVOOH, 2022). S pomocí uvedených funkcionalit lze tyto chyby v evidenci dohledat a opravit. Uživateli jsou k dispozici tři různé varianty oprav.

- „Oprava dvojic“ v případě přesných záznamů provádí opravy pouze v případě, že evidované nesoulady jsou stejné, tj. množství na předání odpovídá množství na převzetí. V případě dalších oprav je nutné provést i tuto opravu, jelikož značně snižuje časovou náročnost dalších oprav a snižuje míru víceznačnosti v systému (např. přesný záznam mohl být použit pro záznam s odchylkou – viz další bod).
- „Oprava dvojic s odchylkou“ umožňuje opravit hmotnostní nesoulady i v případě že si neodpovídají, avšak jsou v toleranci uživatelem zadané hodnoty v červeném poli.
- „Oprava složených záznamů“ vyhledává opravy bez odchylky s tím rozdílem, že vyhledává kombinace vícero záznamů, jejichž součet hmotnostních nesouladů vyrovná zkoumaný záznam. Tento druh opravy nabývá na významu zejména v případě využití agregovaných dat na větší územní celky.

Po stisknutí příslušného tlačítka budou provedeny opravy dle zvolených funkcionalit. Opět je nutné podotknout, že se jedná o časově náročný proces v horizontu jednotek až

desítek minut, zejména pak v případě evidence s větším počtem záznamů a ZÚJ. **Během procesu může MS Excel zdánlivě vypadat, že přestal pracovat. Je nutné patřičnou dobu počkat až do objevení informativního okna o dokončení procesu.** Následující tabulky (viz Obr. 9) poskytují informaci o provedených opravách.



Počet chyb:	Počet
Celkem:	127
Opraveno přesných:	104
Opraveno s odchylkou:	8
Opraveno složených:	15

Chyby Množství:	Množství [t]
Celkem:	121074.96
Opraveno přesných:	93233.75
Opraveno s odchylkou:	9827.45
Opraveno složených:	18013.76

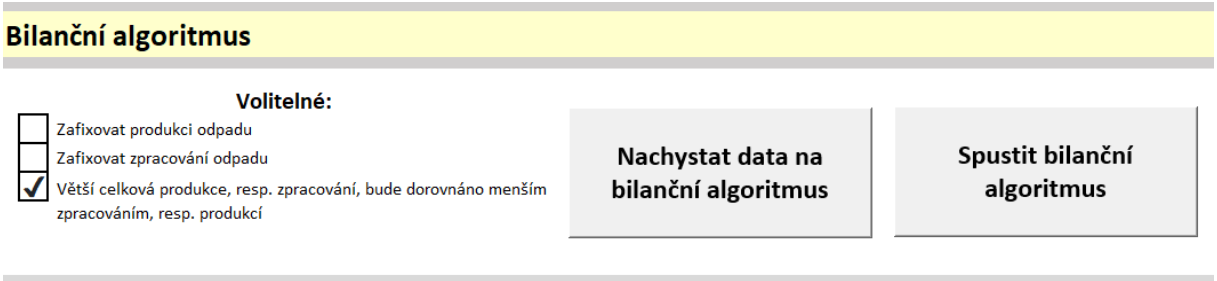
Statistika transportu:	Množství [t]
Odpovídající evidence:	884377
Evidence v nesouladu převzetí:	136815
Evidence v nesouladu předání:	139704
Chybějící převzetí:	24019
Chybějící předání:	43867

Obrázek 9: Sekce „Oprava chyb“ – Informace o provedených opravách.

Červené pole poskytuje informace o počtu opravených záznamů v databázi, zatímco tabulka v zeleném poli je vztažena k opravené hmotnosti. Dílčí položky v tabulkách korespondují s příslušnými funkcionalitami, přičemž první položka uvádí celkový souhrn všech dílčích oprav. K dispozici je také třetí tabulka s informací týkající se chybovosti transportu po opravách. Princip jednotlivých hodnot je stejný jako předchozí sekci „Zpracování dat“.

4.2.4 Sekce „Bilanční algoritmus“

V této části nástroje jsou realizovány poslední přípravy pro bilanční algoritmus v externím modulu. Tato část je spustitelná po provedení alespoň základního zpracování dat, oprava chyb není podmínkou. Nejprve je nutné tlačítkem „Nachystat data na bilanční algoritmus“ (viz Obr. 10) připravit požadovanou strukturu dat. **Proces může být opět časově náročný, je nutné počkat do potvrzující informace o doběhnutí výpočtu.** Dle počtu záznamů a ZÚJ v evidenci může proces trvat jednotky až desítky minut.



Obrázek 10: Sekce „Bilanční algoritmus“.

Následně uživatel může nastavit požadovanou formu vyrovnání, tj. klást důraz na důvěryhodnost vybraných záznamů. Uvedené funckionality jsou volitelné, avšak je doporučené využít alespoň jednu definovanou kombinaci na základě expertního posouzení, přičemž obecně je doporučena třetí varianta. Zároveň je však možné zvolit nejvýše jednu konfiguraci.

- Zafixováním produkce je možné nastavit maximální věrohodnost na evidovanou produkci. Striktně je tak **zakázáno modifikovat produkci** v jakémkoliv ZÚJ a nástroj **upravuje pouze výši zpracování**.
- Zafixováním zpracování je možné nastavit maximální věrohodnost na evidované zpracování. Striktně je tak **zakázáno modifikovat zpracování** v jakémkoliv ZÚJ a nástroj **upravuje pouze výši produkce**.
- Nástroj v rámci bilančního algoritmu je nucen pro splnění hmotnostní bilance systému srovnat celkovou produkci a zpracování. K tomu je možné přistoupit pomocí postupu, kdy nižší hodnota je navýšena alespoň o daný rozdíl. Tento postup reflektuje fakt, že lze očekávat spíše chybějící informace v databázi místo toho, než že by byly uvedeny navíc. Funkcionalita tak zajistí, že **produkce a zpracování bude větší nebo rovno než větší z těchto evidovaných hodnot**. Jednotlivá ZÚJ však nejsou touto podmínkou ovlivněna.

Následně je možné spustit samotný bilanční algoritmus, který je realizován v rámci externího modulu samostatně spustitelného skriptu v jazyce Python v příkazovém řádku. Délka výpočtu je odhadována v řádu desítek vteřin až jednotek minut. Po ukončení procesů v příkazovém řádku jsou k dispozici dva výstupy:

- PDISOH.xlsx – jedná se o novou databázi, která již neobsahuje hmotnostní nesoulady. Některé kódy nakládání jsou zjednodušeny a
- Statistika_Bilance.xlsx – jedná se o podobný formát dat jako na listu „ISOH statistika“ s tím rozdílem, že zde jsou již uvedeny hodnoty po vyrovnání dat. Další rozdíl je v tom, že jednotlivá ZÚJ nemají nakládání rozděleno dle indikátorů, ale dle jednotlivých kódů nakládání. S větším rozsahem dat byl následující agregace na úroveň ORP či krajů umístěny do separátních listů.

4.3 List „DATABAZE“

Tento list obsahuje načtená data z externího souboru. S těmito daty je dále pracováno v dalších analýzách a není doporučeno ručně měnit údaje v tomto listu. Za tímto účelem je list uzamčen. V případě, že uživatel bude potřebovat ručně zasáhnout do načtených dat, případně vložit data ručně bez importu z externího souboru, je možné list příslušným tlačítkem odemknout. Následně však bude nutné provést všechny následující operace týkající se analýz a oprav znovu. Jakmile je provedeno načtení dat, původní struktura v tomto listu bude rozšířena o další sloupce týkající se unikátního ID jednotlivých ZÚJ pro snazší manipulaci s daty.

4.4 List „Výpis chyb“

V případě zakliknutí analýzy chybovosti jsou v tomto listu zobrazeny výsledky analýz. Součástí jsou seznamy chybných ZÚJ dle kontrolovaného kritéria. Vše doplňují souhrnné grafy s vyčíslenou chybovostí a detailní analýza transportních výkazu včetně potřebných popisů jednotlivých výstupů.

4.5 List „ISOH – statistika“

Po zakliknutí jsou v tomto listu ze vstupních dat (tj. bez úprav) vypočteny indikátory nakládání, a další statistiky. Jedná se přesun odpadu mezi kraji, poměrové zpracování a produkce. V tabulkách lze najít také informace k jednotlivým obcím (ZÚJ), které jsou následně agregovány na ORP, Kraje a celou ČR.

4.6 Listy „A – Vlastní odpad“, „B – Cizí odpad“, „C – Skladový odpad“

Obdoba listu “ISOH statistika“ s tím rozdílem, že předmětem analýz jsou pouze kódy nakládání dle příslušného původce. To je možné rozlišit pomocí úvodního písmena:

- Začínající písmenem A, vlastní odpad, tj. jedná se o informace, jak s odpadem nakládají ZÚJ hned po produkci.
- Začínající písmenem B, cizí odpad, tj. jedná se o informace, jak ZÚJ nakládají s odpadem, který převzaly od někoho jiného.
- Začínající písmenem C, odpad z minulého roku, tj. jedná se o informace, jak ZÚJ nakládají s odpadem, který byl uložen na skald v minulém roce.

V případě importování dat z předchozích let jsou zobrazeny také údaje týkající se historického vývoje v minulých letech.

5 Dokumentace – MS Excel

Následující kapitola je věnována podrobnému popisu kódu implementovaném v MS Excel. Struktura kódu je rozdělena do ucelených modulů, které obsahují dílčí spustitelné skripty. Řada skriptů je však navázána na předchozí části nástroje, je proto nutné respektovat příslušnou strukturu daného modulu, který v těchto případech vždy obsahuje jeden souhrnný skript spouštějící postupně ostatní části. Kompletní implementaci kódu lze dohledat v samotném software v MS Excel, karta Vývojář a Visual Basic. V následujících kapitolách budou tedy uvedeny jen popisy funkcionalit příslušných částí kódu.

5.1 Modul „GlobalVariable“

Modul obsahuje definici globálních proměnných, se kterými je nutné pracovat ve více skriptech. Jedná se zejména o počty ZÚJ a dílčí statistické výsledky, které jsou postupně

získávány analýzou vstupních dat. Součástí je také obecná funkce pojmenovaná „IsFile“, která je využívána při načítání dat pro kontrolu existence vstupního souboru.

5.2 Modul „DatabazeLoading“

Tato část kódu je věnována především inicializaci celého nástroje, zejména pak načítání vstupních dat z externích souborů a tvorbě vhodné struktury pro další operace spojené s analýzou dat a přípravou pro vyrovnání dat.

5.2.1 Skript „ButtonNacistVstupy“

Jedná se o řídící skript, který si dle potřeb a nastavení uživatele volá dílčí procedury. Skript zároveň chystá příslušná pole pro výpis a maže staré výpočty.

5.2.2 Skript „AgregaceMestskychCasti“

Tato část kódu je implementována z důvodu potřeb provádět analýzy a výpočty u větších měst rozdělených do dílčích administrativních celků agregovaně. Agregace je provedena na základě skrytého listu „Městské části“, kde jsou uvedeny hledané kódy ZÚJ, které je nutné nahradit.

5.2.3 Skripty „Databaze“ a „SouhrnnaDatabaze“

Skripty slouží k převedení dat z externího souboru, pokud si je uživatel manuálně nevloží do příslušného listu. Skripty jsou rozděleny podle typu vstupního souboru, zda-li se jedná o soubor s jedním katalogovým číslem odpadu, či je v souboru obsaženo více údajů, které s danou analýzou a rekonstrukcí nesouvisí.

5.2.4 Skript „Mnoziny_Uzly“

Cílem této části kódu je získat informaci o počtu dílčích ZÚJ v databázi a vytvořit vhodnou strukturu ID, která poslouží pro rychlejší práci a odkazování na jednotlivé ZÚJ. Zároveň je získána množina obsahující výčet pouze unikátních ZÚJ, která je nutné pro definici optimalizačního modelu.

5.2.5 Skripty „ButtonOdemknoutDatabaze“ a „ButtonZamknoutDatabaze“

Skripty slouží k práci s listem „DATABAZE“, která je ve výchozím stavu uzamčena pro úpravy, aby nemohlo dojít nechtěnou úpravou ke změnám vstupních hodnot mezi dílčími kroky analýz. Kódy slouží tedy k odemčení příslušného listu pro možné modifikace a následně pro jeho opětovné uzamčení.

5.3 Modul „DataAnalysis“

Následující modul obsahuje dílčí skripty zaměřené na samotnou analýzu načtené databáze. Předmětem zkoumaní jsou především hmotnostní bilance jak u jednotlivých ZÚJ, tak i v rámci interakce mezi nimi, které jsou realizovány předáním a převzetím.

5.3.1 Skript „ButtonAnalyzaEvidence“

Jedná se o řídící skript, který na základě uživatelského nastavení v sekci „Zpracování dat“ využívá dílčí části kódu. Některé části jsou však nezbytné pro samotnou přípravu vstupů pro vyrovnání dat a nelze je ovlivnit uživatelskou volbou. Některé volitelné body analýzy jsou součástí až následující modulu 5.4. Procedura současně také připravuje příslušné výpisové tabulky a maže staré hodnoty z dřívějších analýz.

5.3.2 Skript „VypocetBilance“

Pro potřeby vyrovnání dat je nutné načtená data vhodně rozdělit do 4 skupin. Jedná se o produkci, zpracování, předání a převzetí. Dle příslušné definice na skrytém listu „Bilance“ jsou množství odpadů jednotlivých záznamů postupně přiřazována k jednotlivým ZÚJ. Následně je zde implementovaná volitelná analýza předúprav odpadu a vykazování kódu BN40, kdy je porovnáváno množství upravovaného odpadu a sekundární produkce pro výpočet bilance. Další volitelnou funkcionalitou je analýza vykazování skladových zásob, ke které je nutné disponovat evidencí z předchozího roku, než je aktuálně řešený. Při výpočtu této bilance je porovnáno množství odpadu u XN5, XD15 a XR13 v předešlém období s hodnotou záznamu C00 v řešeném roku u každého ZÚJ. Následující části toho skriptu se týkají souhrnných výpisů do připravených polí listu „MENU“, tak i do dalších skrytých listů, odkud je s výsledky pracováno při dalších výpočtech.

5.3.3 Skripty „Produkce“ a „Zpracovani“

Výstupem těchto skriptů je příprava dat o produkci a zpracování odpadu do příslušné struktury pro pozdější export dat do optimalizačního nástroje obsahující bilanční algoritmus založený na vyrovnání dat.

5.3.4 Skripty „Toky_Plus“ a „Toky_Minus“

Jedná se o přípravu dat pro následující skript, kde bude detailně analyzována hmotnostní bilance v rámci všech evidovaných předání a převzetí. Výstupem procedur je informace o toku (hraně) odpadu s jasně definovaným odesílatelem a příjemcem včetně evidovaného množství odpadu jak na straně převzetí („Toky_Plus“), tak na straně předání („Toky_Minus“)

5.3.5 Skript „AnalyzaTransportu“

Díky předchozímu skriptu a vhodně nachystané struktuře dat je možné realizovat porovnání jednotlivých toků. Skript tedy páruje výkazy na straně předání a převzetí a hodnoty porovnává z pohledu hmotnostní bilance. Výsledný nesoulad je následně ukládán pro další účely softwaru.

5.4 Modul „ISOHStatistics“

Modul je zaměřen na analýzu archivní databáze (načtená data) dle preferencí uživatele v sekci „Zpracování dat“ na listu „MENU“. Součástí je vyhodnocení indikátorů odpadového hospodářství (MZP, 2022), dílčí vyhodnocení dle původu odpadu a vyhodnocení vývoje v posledních třech letech. Řízení dílčích kódů je v kompetenci skriptu „ButtonAnalyzaEvidence“, aby veškeré požadované operace proběhly najednou.

5.4.1 Skript „AnalyzaEvidence“

Cílem skriptu je detailní analýza evidence z pohledu všech ZÚJ, která jsou následně agregována na úroveň ORP, krajů a celé ČR. Výsledkem jsou informace týkající se produkce a vyhodnocení indikátorů nakládání s odpady dle příslušných kritérií. Součástí je také analýza přepravy odpadu mezi kraji.

5.4.2 Skript „Statistika_PuvodceOdpadu“

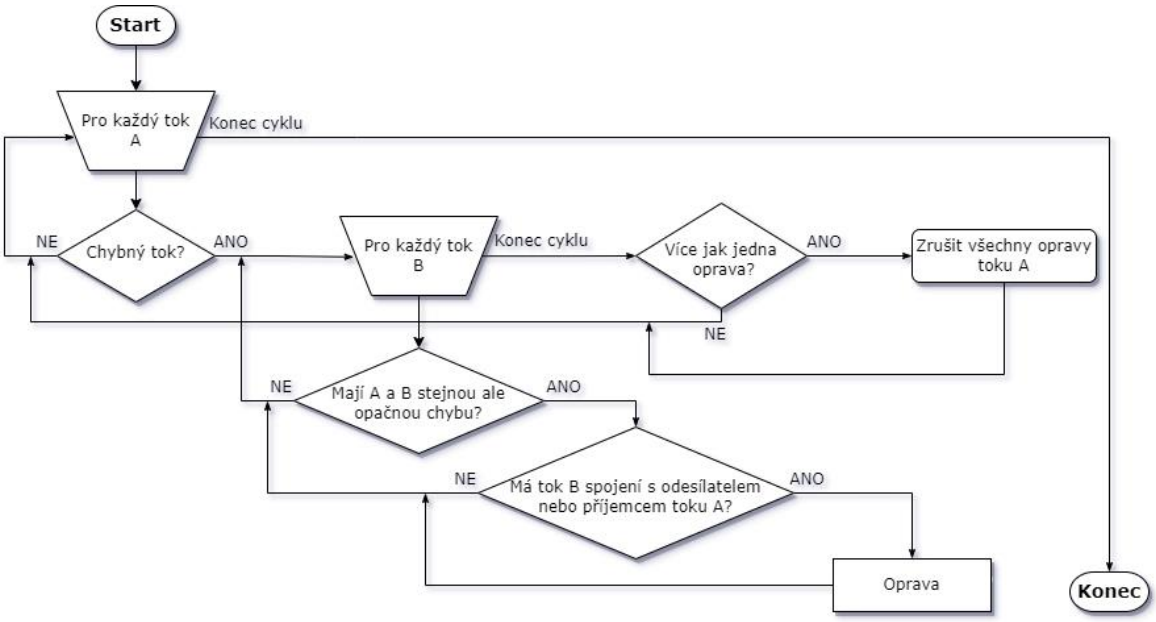
Tento skript provádí stejnou analýzu jako předchozí skript „AnalyzaEvidence“ s tím rozdílem, že odpad dále dělí dle kategorie původce daného kódu nakládání (A/B/C). V případě, že jsou k dispozici i data z předchozích roků, je součástí analýzy také vývoj za poslední tři roky.

5.4.3 Skript „ChybyZUJ“

V rámci tohoto skriptu jsou vypisovány vypočítané statistiky chybovosti do příslušného listu. Rozsah analýz je definován samotným uživatelem pomocí volitelných funkcionalit v sekce „Zpracování dat“.

5.5 Modul „TransactionFixing“

Modul je zaměřen na algoritmické opravování transakcí spojené s předáním a převzetím odpadu. Základní myšlenka oprav je založena na porovnávání hmotnostních nesouladů na jednotlivých hranách (tocích mezi dvěma ZÚJ), tj. pro každou hranu je definovaná bilance jako rozdíl výkazu převzetí a předání. Pokud po sečtení dvou vybraných toků s nesouladem je bilance nulová, je pravděpodobné, že tyto toky mohou být navzájem opraveny. Podmínkou pro opravu je, že oba toky musí spojovat jedno ZÚJ. Algoritmus pro opravu je zobrazen na Obr. 11.



Obrázek 11: Vývojový diagram popisující opravu chyb při přepravě.

5.5.1 Skript „ButtonOpravyChyb“

Jedná se o řídící skript celé opravy, který postupně dle navolených funkcionalit provádí požadované operace. Současně jsou připravovány výpisové pole v listu „MENU“.

5.5.2 Skript „NajitDvojicePresne“

Výše uvedeny opravný algoritmus je v tomto skriptu využit pro hledání takových toků k opravě, u kterých si je hmotnostní nesoulad v absolutní hodnotě roven.

5.5.3 Skript „NajitDvojiceOdchylka“

Tento skript provádí podobnou operaci jako předchozí s tím rozdílem, že připouští i opravy do uživatelem zadané absolutní odchylky.

5.5.4 Skript „OpravaDvojice“

Skript opraví nalezené možné opravy a zohlední je do všech statistik a následujících vyhodnocení.

5.5.5 Skript „NajitKombinace“

Skript na rozdíl od přechozího hledání pouhých dvojic k opravě hledá i možné kombinace vícero toků najednou. Tj. hmotnostní nesoulad jednoho toku odpadu mezi dvěma ZÚJ může být opraven s využitím dvou či více toků. Podmínkou ovšem stále zůstává, že všechny toky musí mít jedno společné ZÚJ.

5.5.6 Skript „OpravaKombinace“

Následující skript opravuje nalezené návrhy k opravě skládajících se z vícero záznamů najednou.

5.6 Modul „Reconciliation“

Tento modul realizuje přípravu na vyrovnání dat, vypočítává váhy pro jednotlivá ZÚJ v evidenci a všechna potřebná data umisťuje do regionů, na které se následně odkazuje externí výpočtový modul v jazyce Python.

5.6.1 Skript „ButtonNachystejVahy“

Jedná se o řídící skript, který postupně spouští dílčí skripty týkající se vah jednotlivých ZÚJ a toků odpadů mezi nimi.

5.6.2 Skript „ButtonSpustitBilanci“

Jedná se o řídící skript, který kontroluje, zda-li je vše korektně nachystáno na výpočet a následně s pomocí příkazového řádku spustí externí výpočtový modul.

5.6.3 Skript „ChybovostTransportuUzel“

Skript napočítává chybovost jednotlivých ZÚJ v předání a převzetí, tj. výstupem je hodnotící kritérium každého ZÚJ v evidenci, vyjadřující míru nesouladů při vykazování transportu s ostatními ZÚJ.

5.6.4 Skripty „VahyProdukce“ a „VahyZpracovani“

Skripty vytvoří hodnotící kritérium jak pro produkční část ZÚJ, tak pro zpracovatelskou část. Postupně jsou zohledňovány různé aspekty dle metodiky M1F2 k V1.F.2.1 (CEVOOH, 2022)

5.6.5 Skript „IncMatice“

Skript zajišťuje vytvoření vstupních parametrů týkajících se předání a převzetí, tj. toků odpadu mezi ZÚJ, pro externí výpočtový modul. Součástí je incidenční matice, která popisuje návaznost hran a uzlů, ve výsledku tedy vyjadřuje strukturu síťového grafu.

5.7 List „List6 (MENU)“

Pro vylepšení uživatelského rozhraní ve formě zaklikávacích políček bylo nutné implementovat příslušný kód do objektu navázaného na daný list. Jedná se však pouze o vizuální aspekt popisovaného softwaru.

6 Dokumentace – Python

Optimalizační model založený na vyrovnání dat je nutné mít externě, jelikož MS Excel neposkytuje potřebné nástroje, případně není tak efektivní, což by vzhledem k rozsahu řešené úlohy mohlo být značně omezující. Následující část dokumentace bude podrobně popisovat funkcionality jednotlivých částí kódů včetně příkladů implementace.

6.1 Využité knihovny

Jako první je nutné importovat potřebné knihovny, které usnadní práci s datovými strukturami či nabízejí komplexní výpočtové nástroje. Výčet využitých knihoven je následující:

- **Pandas** – knihovna poskytuje výkonné nástroje pro import, manipulaci, analýzu a vizualizaci dat. Pandas umožňuje snadno načítat a ukládat data ve formě tabulek, které se nazývají DataFrames. Tato knihovna nabízí širokou škálu funkcí pro filtrování, transformaci a spojování dat, což usnadňuje práci s velkými datovými soubory.
- **Numpy** – knihovna poskytuje výkonné nástroje pro práci s numerickými daty v Pythonu. Je základním nástrojem pro vědecké výpočty a umožňuje efektivní manipulaci s maticemi a vektory.
- **Openpyxl** – knihovna umožňuje čtení a zápis dat do souborů programu Microsoft Excel pomocí Pythonu. Díky tomuto rozhraní je možno snadno manipulovat s listy, buňkami, tabulkami a grafy v Excelu.
- **Pathlib** – knihovna poskytuje jednoduché a přenositelné rozhraní pro manipulaci s cestami a soubory v souborovém systému. Pathlib je navržena tak, aby byla platformě nezávislá, což znamená, že lze používat stejné kódy na různých operačních systémech (Windows, Linux, macOS) bez nutnosti psát specifický kód pro každý systém.
- **Pulp** – knihovna slouží k řešení optimalizačních problémů, zejména lineárních. Poskytuje jednoduché a intuitivní rozhraní pro definování a řešení matematických modelů. Pulp je užitečným nástrojem pro modelování a řešení komplexních optimalizačních problémů, které se vyskytují v různých odvětvích a disciplínách.

6.2 Funkce „data_frame_from_xlsx“

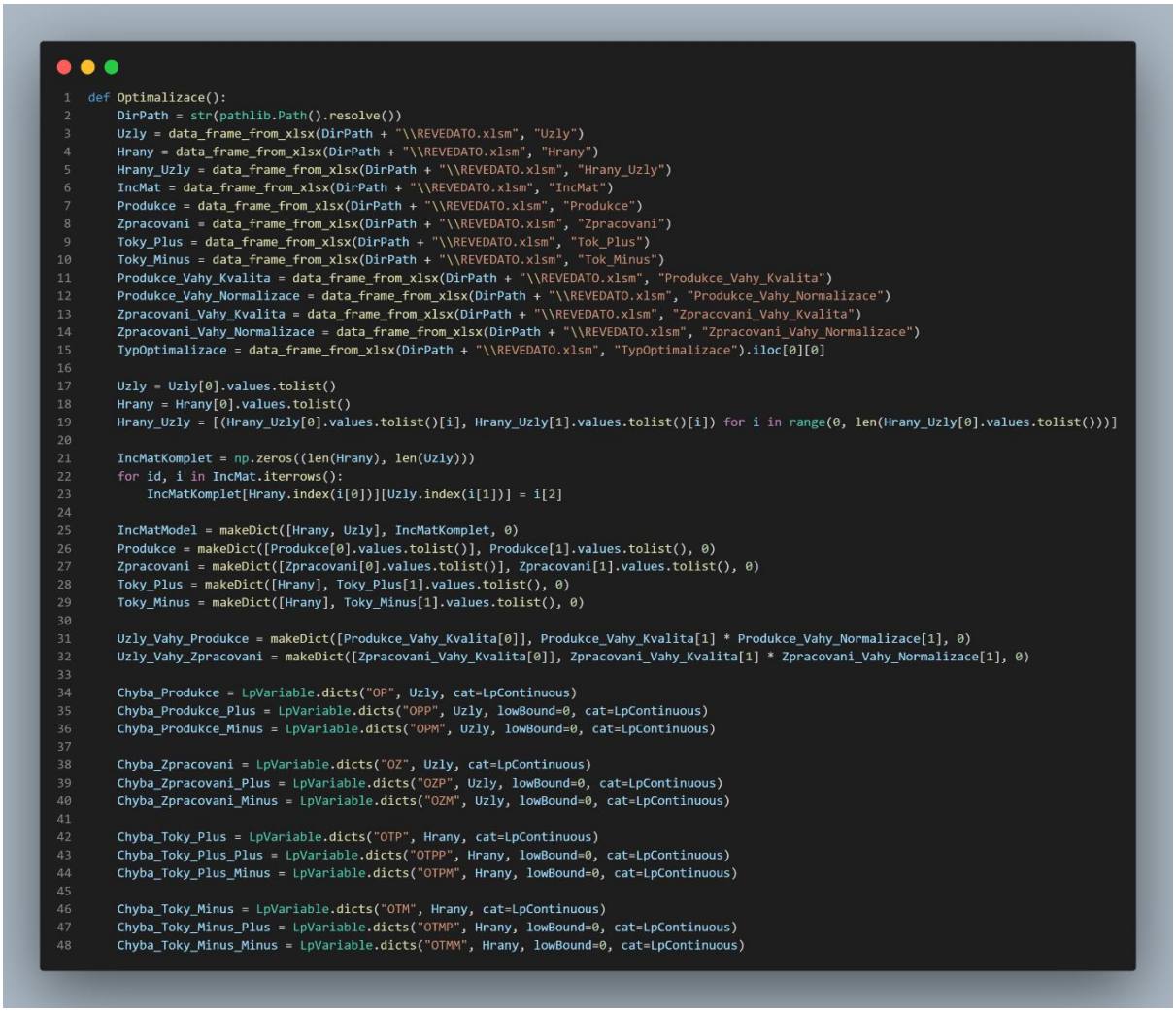
Tato funkce umožňuje importovat data z MS Excel, konkrétně poskytuje rozhraní pro načtení dat přímo z pojmenovaných regionů. Tento přístup je výhodný zejména kvůli upořádání dat v MS Excel a umožňuje tak mít všechna potřebná data v jednom souboru. Na Obr. 12 je k dispozici samotná implementace využívající knihoven **openpyxl** a **pandas**.

```python
def data_frame_from_xlsx(xlsx_file, range_name):
    wb = openpyxl.load_workbook(xlsx_file, data_only=True, read_only=True)
    full_range = wb.defined_names[range_name]
    if full_range is None:
        raise ValueError(
            'Range "{}" not found in workbook "{}".'.format(range_name, xlsx_file)
        )
    destinations = list(full_range.destinations)
    if len(destinations) > 1:
        raise ValueError(
            'Range "{}" in workbook "{}" contains more than one region.'
            .format(range_name, xlsx_file)
        )
    ws, reg = destinations[0]
    if isinstance(ws, str):
        ws = wb[ws]
    region = ws[reg]

    df = pd.DataFrame([cell.value for cell in row] for row in region)
    return df
```

Obrázek 12: Implementace funkce „data_frame_from_xlsx“.

6.3 Funkce „Optimalizace“

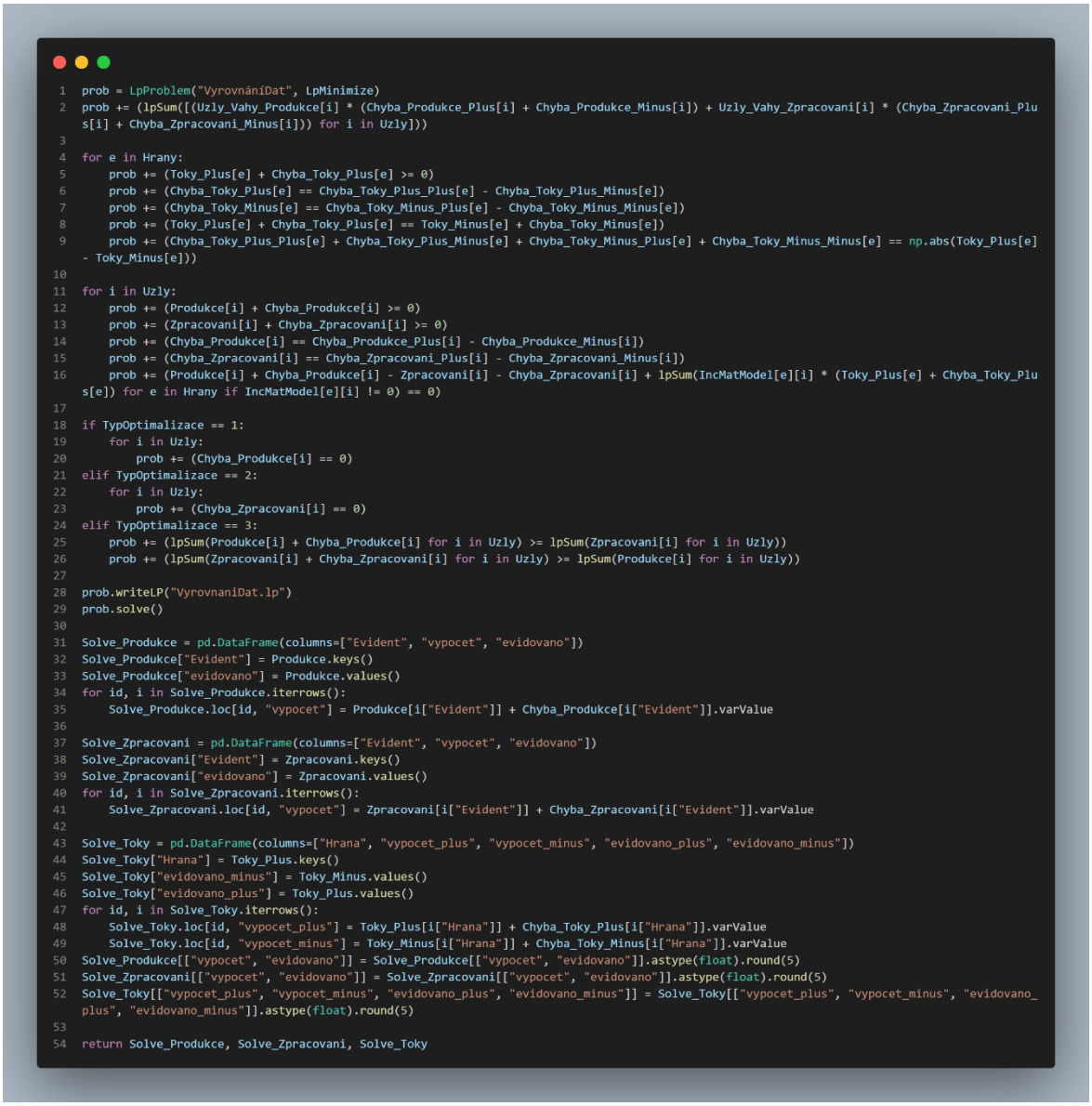
Jádro externího modulu je právě funkce „Optimalizace“, která s využitím optimalizačních knihoven obsahuje naimplementované vyrovnání dat pro rekonstrukci toků. V první řadě je však nutné provést import všech nachystaných dat v průběhu dílčích analýz a výpočtů v MS Excel. Tento import je realizován s využitím předchozí funkce „data_frame_from_xlsx“. Následně je nutné připravit potřebné množiny, parametry a proměnné pro model. Množiny je nutné z formátu „DataFrame“ transformovat na strukturu „List“, parametry s využitím funkce „makeDict“ na strukturu „Dict“, kde klíčem jsou definované množiny. Proměnné jsou definovány s pomocí funkce „LpVariable.dicts“ a požadovaných parametrů jako typ spojitosti či specifické omezení jako nezápornost. Implementace je vizualizována na Obr. 13.



```python
def Optimalizace():
    DirPath = str(pathlib.Path().resolve())
    Uzly = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Uzly")
    Hrany = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Hrany")
    Hrany_Uzly = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Hrany_Uzly")
    IncMat = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "IncMat")
    Produkce = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Produkce")
    Zpracovani = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Zpracovani")
    Toky_Plus = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Tok_Plus")
    Toky_Minus = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Tok_Minus")
    Produkce_Vahy_Kvalita = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Produkce_Vahy_Kvalita")
    Produkce_Vahy_Normalizace = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Produkce_Vahy_Normalizace")
    Zpracovani_Vahy_Kvalita = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Zpracovani_Vahy_Kvalita")
    Zpracovani_Vahy_Normalizace = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Zpracovani_Vahy_Normalizace")
    TypOptimalizace = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "TypOptimalizace").iloc[0][0]

    Uzly = Uzly[0].values.tolist()
    Hrany = Hrany[0].values.tolist()
    Hrany_Uzly = [(Hrany_Uzly[0].values.tolist()[i], Hrany_Uzly[1].values.tolist()[i]) for i in range(0, len(Hrany_Uzly[0].values.tolist()))]

    IncMatKomplet = np.zeros((len(Hrany), len(Uzly)))
    for id, i in IncMat.iterrows():
        IncMatKomplet[Hrany.index(i[0])][Uzly.index(i[1])] = i[2]

    IncMatModel = makeDict([Hrany, Uzly], IncMatKomplet, 0)
    Produkce = makeDict([Produkce[0].values.tolist()], Produkce[1].values.tolist(), 0)
    Zpracovani = makeDict([Zpracovani[0].values.tolist()], Zpracovani[1].values.tolist(), 0)
    Toky_Plus = makeDict([Hrany], Toky_Plus[1].values.tolist(), 0)
    Toky_Minus = makeDict([Hrany], Toky_Minus[1].values.tolist(), 0)

    Uzly_Vahy_Produkce = makeDict([Produkce_Vahy_Kvalita[0]], Produkce_Vahy_Kvalita[1] * Produkce_Vahy_Normalizace[1], 0)
    Uzly_Vahy_Zpracovani = makeDict([Zpracovani_Vahy_Kvalita[0]], Zpracovani_Vahy_Kvalita[1] * Zpracovani_Vahy_Normalizace[1], 0)

    Chyba_Produkce = LpVariable.dicts("OP", Uzly, cat=LpContinuous)
    Chyba_Produkce_Plus = LpVariable.dicts("OPP", Uzly, lowBound=0, cat=LpContinuous)
    Chyba_Produkce_Minus = LpVariable.dicts("OPM", Uzly, lowBound=0, cat=LpContinuous)

    Chyba_Zpracovani = LpVariable.dicts("OZ", Uzly, cat=LpContinuous)
    Chyba_Zpracovani_Plus = LpVariable.dicts("OZP", Uzly, lowBound=0, cat=LpContinuous)
    Chyba_Zpracovani_Minus = LpVariable.dicts("OZM", Uzly, lowBound=0, cat=LpContinuous)

    Chyba_Toky_Plus = LpVariable.dicts("OTP", Hrany, cat=LpContinuous)
    Chyba_Toky_Plus_Plus = LpVariable.dicts("OTPP", Hrany, lowBound=0, cat=LpContinuous)
    Chyba_Toky_Plus_Minus = LpVariable.dicts("OTPM", Hrany, lowBound=0, cat=LpContinuous)

    Chyba_Toky_Minus = LpVariable.dicts("OTM", Hrany, cat=LpContinuous)
    Chyba_Toky_Minus_Plus = LpVariable.dicts("OTMP", Hrany, lowBound=0, cat=LpContinuous)
    Chyba_Toky_Minus_Minus = LpVariable.dicts("OTMM", Hrany, lowBound=0, cat=LpContinuous)
```

Obrázek 13: Implementace funkce „Optimalizace“ – Import dat, příprava množin, parametrů a proměnných pro optimalizační model.

Následně je možné přistoupit k samotné definici optimalizačního modelu sestávající se z účelové funkce a sady omezení. Inicializace modelu je realizováno s využitím funkce „LpProblem“. Poté je definována účelová funkce, omezení pro každou hranu a uzel. Příslušné matematické vyjádření je ekvivalentní s dokumentem XX. Optimalizační výpočet je spuštěn s využitím „writeLP“ a „solve“. Jelikož proměnné jsou definovány jako odchylky od příslušných parametrů, je nutné po výpočtu výsledky rekonstrukce, tj. hodnoty proměnných, zohlednit do vstupních dat a připravit do vhodné struktury „DataFrame“. Implementace kódu je k vidění na Obr. 14.



```python
prob = LpProblem("VyrovnáníDat", LpMinimize)
prob += (lpSum([(Uzly_Vahy_Produkce[i] * (Chyba_Produkce_Plus[i] + Chyba_Produkce_Minus[i]) + Uzly_Vahy_Zpracovani[i] * (Chyba_Zpracovani_Plus[i] + Chyba_Zpracovani_Minus[i])) for i in Uzly]))

for e in Hrany:
    prob += (Toky_Plus[e] + Chyba_Toky_Plus[e] >= 0)
    prob += (Chyba_Toky_Plus[e] == Chyba_Toky_Plus_Plus[e] - Chyba_Toky_Plus_Minus[e])
    prob += (Chyba_Toky_Minus[e] == Chyba_Toky_Minus_Plus[e] - Chyba_Toky_Minus_Minus[e])
    prob += (Toky_Plus[e] + Chyba_Toky_Plus[e] == Toky_Minus[e] + Chyba_Toky_Minus[e])
    prob += (Chyba_Toky_Plus_Plus[e] + Chyba_Toky_Plus_Minus[e] + Chyba_Toky_Minus_Plus[e] + Chyba_Toky_Minus_Minus[e] == np.abs(Toky_Plus[e] - Toky_Minus[e]))

for i in Uzly:
    prob += (Produkce[i] + Chyba_Produkce[i] >= 0)
    prob += (Zpracovani[i] + Chyba_Zpracovani[i] >= 0)
    prob += (Chyba_Produkce[i] == Chyba_Produkce_Plus[i] - Chyba_Produkce_Minus[i])
    prob += (Chyba_Zpracovani[i] == Chyba_Zpracovani_Plus[i] - Chyba_Zpracovani_Minus[i])
    prob += (Produkce[i] + Chyba_Produkce[i] - Zpracovani[i] - Chyba_Zpracovani[i] + lpSum(IncMatModel[e][i] * (Toky_Plus[e] + Chyba_Toky_Plus[e]) for e in Hrany if IncMatModel[e][i] != 0) == 0)

if TypOptimalizace == 1:
    for i in Uzly:
        prob += (Chyba_Produkce[i] == 0)
elif TypOptimalizace == 2:
    for i in Uzly:
        prob += (Chyba_Zpracovani[i] == 0)
elif TypOptimalizace == 3:
    prob += (lpSum(Produkce[i] + Chyba_Produkce[i] for i in Uzly) >= lpSum(Zpracovani[i] for i in Uzly))
    prob += (lpSum(Zpracovani[i] + Chyba_Zpracovani[i] for i in Uzly) >= lpSum(Produkce[i] for i in Uzly))

prob.writeLP("VyrovnaniDat.lp")
prob.solve()

Solve_Produkce = pd.DataFrame(columns=["Evident", "vypocet", "evidovano"])
Solve_Produkce["Evident"] = Produkce.keys()
Solve_Produkce["evidovano"] = Produkce.values()
for id, i in Solve_Produkce.iterrows():
    Solve_Produkce.loc[id, "vypocet"] = Produkce[i["Evident"]] + Chyba_Produkce[i["Evident"]].varValue

Solve_Zpracovani = pd.DataFrame(columns=["Evident", "vypocet", "evidovano"])
Solve_Zpracovani["Evident"] = Zpracovani.keys()
Solve_Zpracovani["evidovano"] = Zpracovani.values()
for id, i in Solve_Zpracovani.iterrows():
    Solve_Zpracovani.loc[id, "vypocet"] = Zpracovani[i["Evident"]] + Chyba_Zpracovani[i["Evident"]].varValue

Solve_Toky = pd.DataFrame(columns=["Hrana", "vypocet_plus", "vypocet_minus", "evidovano_plus", "evidovano_minus"])
Solve_Toky["Hrana"] = Toky_Plus.keys()
Solve_Toky["evidovano_minus"] = Toky_Minus.values()
Solve_Toky["evidovano_plus"] = Toky_Plus.values()
for id, i in Solve_Toky.iterrows():
    Solve_Toky.loc[id, "vypocet_plus"] = Toky_Plus[i["Hrana"]] + Chyba_Toky_Plus[i["Hrana"]].varValue
    Solve_Toky.loc[id, "vypocet_minus"] = Toky_Minus[i["Hrana"]] + Chyba_Toky_Minus[i["Hrana"]].varValue
Solve_Produkce[["vypocet", "evidovano"]] = Solve_Produkce[["vypocet", "evidovano"]].astype(float).round(5)
Solve_Zpracovani[["vypocet", "evidovano"]] = Solve_Zpracovani[["vypocet", "evidovano"]].astype(float).round(5)
Solve_Toky[["vypocet_plus", "vypocet_minus", "evidovano_plus", "evidovano_minus"]] = Solve_Toky[["vypocet_plus", "vypocet_minus", "evidovano_plus", "evidovano_minus"]].astype(float).round(5)

return Solve_Produkce, Solve_Zpracovani, Solve_Toky
```

Obrázek 14: Implementace funkce „Optimalizace“ – Definice optimalizačního modelu a příprava vhodného formátu pro výstup.

6.4 Funkce „Statistika“

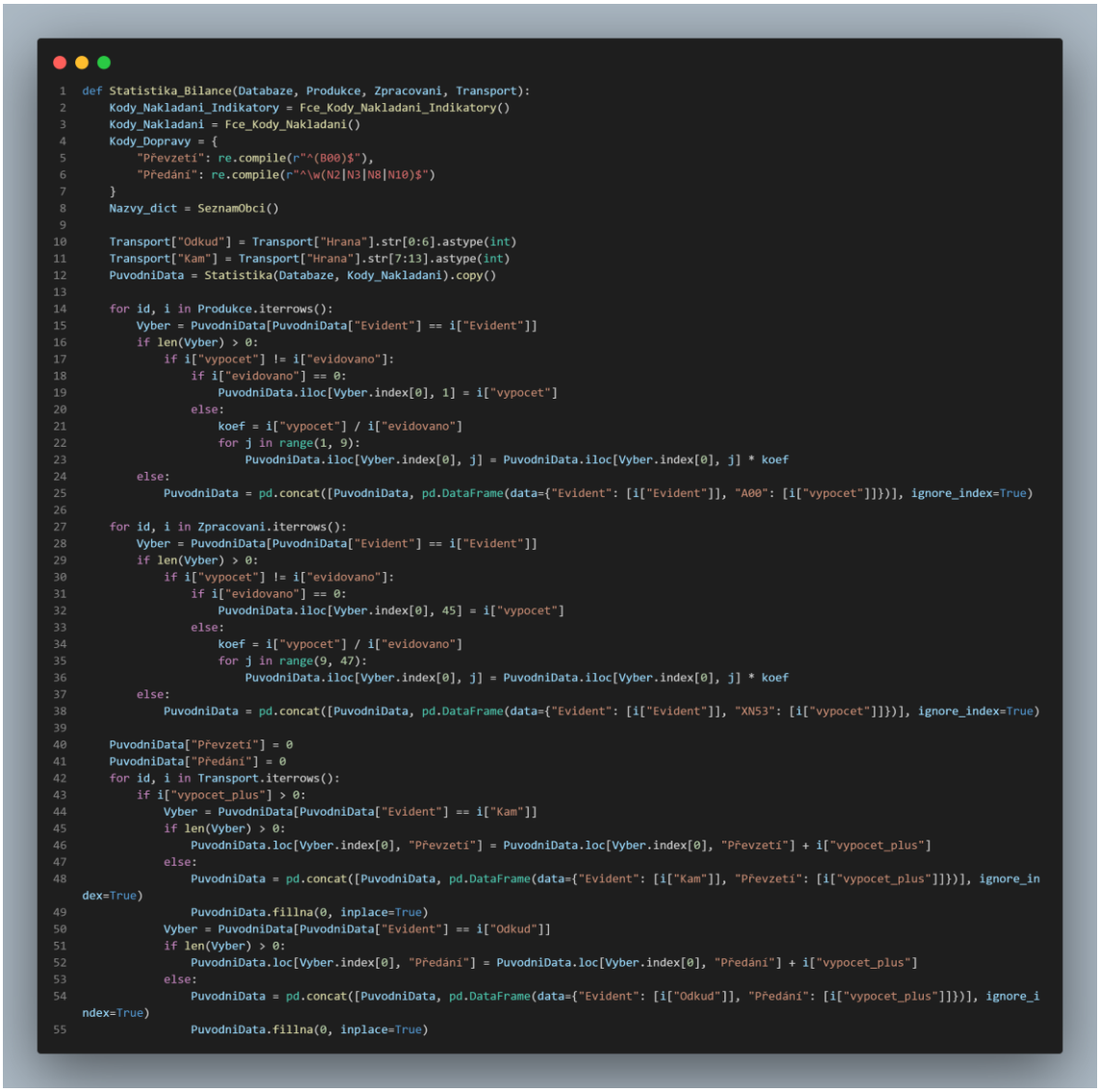
Tato funkce zajišťuje vytvoření tabulkové struktury z původních dat určených k rekonstrukci. Řádky zastupují jednotlivé ZÚJ, zatímco sloupce kódy nakládání. Z původní databáze jsou pak přiřazovány hodnoty dle reportovaných záznamů. Tento výstup je následně využit jako základ pro zohlednění odchylek z výpočtu. Příslušná implementace je na Obr. 15.

```
def Statistika(Databaze, Kody_Nakladani):
    Databaze.fillna(0, inplace=True)
    Databaze["Množství"] = Databaze["Množství odpadu + (t)"] + Databaze["Množství odpadu - (t)"]
    Databaze = Databaze[["Evident", "Kód nakládání", "Množství"]]
    Output = Databaze["Evident"].drop_duplicates().reset_index(drop=True)
    for key in Kody_Nakladani:
        Nakladani = Databaze.loc[Databaze["Kód nakládání"].str.match(Kody_Nakladani[key])].copy()
        Nakladani[key] = Nakladani.groupby(["Evident"])["Množství"].transform("sum")
        Nakladani = Nakladani.drop_duplicates(subset=["Evident"]).reset_index(drop=True)
        Output = pd.merge(Output, Nakladani[["Evident", key]], how="outer", on=["Evident"])
    Output.fillna(0, inplace=True)
    Output = Output.round(5)
    return Output
```

Obrázek 15: Implementace funkce „Statistika“.

6.5 Funkce „Statistika_Bilance“

Funkce zajišťuje zohlednění vypočtených odchylek od vstupních parametrů do původních dat evidence. Odchylky jsou do jednotlivých kódů nakládání rozděleny poměrově dle velikost vykázaných hodnot, tj. nulové hodnoty nejsou ovlivněny, v případě pouze jednoho výkazu se veškerá změna projeví jen v jednom záznamu. Příslušná implementace je k dispozici na Obr. 16.



```python
def Statistika_Bilance(Databaze, Produkce, Zpracovani, Transport):
    Kody_Nakladani_Indikatory = Fce_Kody_Nakladani_Indikatory()
    Kody_Nakladani = Fce_Kody_Nakladani()
    Kody_Dopravy = {
        "Převzetí": re.compile(r"^(B00)$"),
        "Předání": re.compile(r"^\w(N2|N3|N8|N10)$")
    }
    Nazvy_dict = SeznamObci()

    Transport["Odkud"] = Transport["Hrana"].str[0:6].astype(int)
    Transport["Kam"] = Transport["Hrana"].str[7:13].astype(int)
    PuvodniData = Statistika(Databaze, Kody_Nakladani).copy()

    for id, i in Produkce.iterrows():
        Vyber = PuvodniData[PuvodniData["Evident"] == i["Evident"]]
        if len(Vyber) > 0:
            if i["vypocet"] != i["evidovano"]:
                if i["evidovano"] == 0:
                    PuvodniData.iloc[Vyber.index[0], 1] = i["vypocet"]
                else:
                    koef = i["vypocet"] / i["evidovano"]
                    for j in range(1, 9):
                        PuvodniData.iloc[Vyber.index[0], j] = PuvodniData.iloc[Vyber.index[0], j] * koef
        else:
            PuvodniData = pd.concat([PuvodniData, pd.DataFrame(data={"Evident": [i["Evident"]], "A00": [i["vypocet"]]})], ignore_index=True)

    for id, i in Zpracovani.iterrows():
        Vyber = PuvodniData[PuvodniData["Evident"] == i["Evident"]]
        if len(Vyber) > 0:
            if i["vypocet"] != i["evidovano"]:
                if i["evidovano"] == 0:
                    PuvodniData.iloc[Vyber.index[0], 45] = i["vypocet"]
                else:
                    koef = i["vypocet"] / i["evidovano"]
                    for j in range(9, 47):
                        PuvodniData.iloc[Vyber.index[0], j] = PuvodniData.iloc[Vyber.index[0], j] * koef
        else:
            PuvodniData = pd.concat([PuvodniData, pd.DataFrame(data={"Evident": [i["Evident"]], "XN53": [i["vypocet"]]})], ignore_index=True)

    PuvodniData["Převzetí"] = 0
    PuvodniData["Předání"] = 0
    for id, i in Transport.iterrows():
        if i["vypocet_plus"] > 0:
            Vyber = PuvodniData[PuvodniData["Evident"] == i["Kam"]]
            if len(Vyber) > 0:
                PuvodniData.loc[Vyber.index[0], "Převzetí"] = PuvodniData.loc[Vyber.index[0], "Převzetí"] + i["vypocet_plus"]
            else:
                PuvodniData = pd.concat([PuvodniData, pd.DataFrame(data={"Evident": [i["Kam"]], "Převzetí": [i["vypocet_plus"]]})], ignore_in
dex=True)
                PuvodniData.fillna(0, inplace=True)
            Vyber = PuvodniData[PuvodniData["Evident"] == i["Odkud"]]
            if len(Vyber) > 0:
                PuvodniData.loc[Vyber.index[0], "Předání"] = PuvodniData.loc[Vyber.index[0], "Předání"] + i["vypocet_plus"]
            else:
                PuvodniData = pd.concat([PuvodniData, pd.DataFrame(data={"Evident": [i["Odkud"]], "Předání": [i["vypocet_plus"]]})], ignore_i
ndex=True)
                PuvodniData.fillna(0, inplace=True)
```

Obrázek 16: Implementace funkce „Statistika_Bilance“ – Zohlednění výsledků rekonstrukce do původní databáze.

Další část kódu této funkce se zaměřuje na formátování výpisu. V rámci výstupního souboru „Statistika_Bilance.xlsx“ je vytvořen přehled za všechna ZÚJ, která jsou následně na dalších listech agregována na úroveň ORP a krajů. Druhý výstupní soubor „PDISOH.xlsx“ obsahuje strukturu podobnou původním vstupním datům, tj. jedná se o jednotlivé záznamy dílčích ZÚJ, které však mohou být agregovány v případě, že ZÚJ vykázalo dva záznamy pod stejným kódem nakládání a stejným partnerem. Implementace kódu je na Obr. 17.

```python
PuvodniData.fillna(0, inplace=True)
PuvodniData.insert(1, "Evident - Typ subjektu", "")
PuvodniData.insert(2, "Evident - Nazev", "")
PuvodniData.insert(3, "Evident - Kraj", "")
PuvodniData["Evident - Typ subjektu"] = 0
for id, i in PuvodniData.iterrows():
    PuvodniData.loc[id, "Evident - Nazev"] = Nazvy_dict[int(PuvodniData.loc[id, "Evident"])][0]
    PuvodniData.loc[id, "Evident - Kraj"] = Nazvy_dict[int(PuvodniData.loc[id, "Evident"])][1]

PuvodniData["Partner"] = PuvodniData["Evident"]
PuvodniData["Partner - Typ subjektu"] = PuvodniData["Evident - Typ subjektu"]
PuvodniData["Partner - Nazev"] = PuvodniData["Evident - Nazev"]
PuvodniData["Partner - Kraj"] = PuvodniData["Evident - Kraj"]
PracovniDatabaze = pd.DataFrame(columns=["Evident", "Evident - Typ subjektu", "Evident - Nazev", "Evident - Kraj", "Partner", "Partner - Typ subjektu", "Partner - Nazev", "Partner - Kraj", "Kod", "Indikator", "Mnozstvi"])

for i in Kody_Nakladani:
    if i not in Kody_Dopravy:
        PuvodniData["Kod"] = i
        for j in Kody_Nakladani_Indikatory:
            if Kody_Nakladani_Indikatory[j].match(i):
                PuvodniData["Indikator"] = j
                break
        PracovniDatabaze = pd.concat([PracovniDatabaze, PuvodniData[PuvodniData[i] > 0][["Evident", "Evident - Typ subjektu", "Evident - Nazev", "Evident - Kraj", "Partner", "Partner - Typ subjektu", "Partner - Nazev", "Partner - Kraj", "Kod", "Indikator", i]].rename(columns={i: "Mnozstvi"})], ignore_index=True)[PracovniDatabaze.columns]

Transport.insert(0, "Evident - Nazev", "")
Transport.insert(0, "Evident - Kraj", "")
Transport.insert(0, "Partner - Nazev", "")
Transport.insert(0, "Partner - Kraj", "")
Transport["Kod"] = "Předání"
Transport["Indikator"] = "Předání"
Transport["Evident"] = Transport["Odkud"]
Transport["Evident - Typ subjektu"] = 0
Transport["Partner"] = Transport["Kam"]
Transport["Partner - Typ subjektu"] = 0
for id, i in Transport.iterrows():
    Transport.loc[id, "Evident - Nazev"] = Nazvy_dict[int(Transport.loc[id, "Evident"])][0]
    Transport.loc[id, "Evident - Kraj"] = Nazvy_dict[int(Transport.loc[id, "Evident"])][1]
    Transport.loc[id, "Partner - Nazev"] = Nazvy_dict[int(Transport.loc[id, "Partner"])][0]
    Transport.loc[id, "Partner - Kraj"] = Nazvy_dict[int(Transport.loc[id, "Partner"])][1]
PracovniDatabaze = pd.concat([PracovniDatabaze, Transport[Transport["vypocet_plus"] > 0][["Evident", "Evident - Typ subjektu", "Evident - Nazev", "Evident - Kraj", "Partner", "Partner - Typ subjektu", "Partner - Nazev", "Partner - Kraj", "Kod", "Indikator", "vypocet_plus"]].rename(columns={"vypocet_plus": "Mnozstvi"})], ignore_index=True)[PracovniDatabaze.columns]

Transport["Kod"] = "Převzetí"
Transport["Indikator"] = "Převzetí"
Transport["Evident"] = Transport["Kam"]
Transport["Evident - Typ subjektu"] = 0
Transport["Partner"] = Transport["Odkud"]
Transport["Partner - Typ subjektu"] = 0
for id, i in Transport.iterrows():
    Transport.loc[id, "Evident - Nazev"] = Nazvy_dict[int(Transport.loc[id, "Evident"])][0]
    Transport.loc[id, "Evident - Kraj"] = Nazvy_dict[int(Transport.loc[id, "Evident"])][1]
    Transport.loc[id, "Partner - Nazev"] = Nazvy_dict[int(Transport.loc[id, "Partner"])][0]
    Transport.loc[id, "Partner - Kraj"] = Nazvy_dict[int(Transport.loc[id, "Partner"])][1]
PracovniDatabaze = pd.concat([PracovniDatabaze, Transport[Transport["vypocet_plus"] > 0][["Evident", "Evident - Typ subjektu", "Evident - Nazev", "Evident - Kraj", "Partner", "Partner - Typ subjektu", "Partner - Nazev", "Partner - Kraj", "Kod", "Indikator", "vypocet_plus"]].rename(columns={"vypocet_plus": "Mnozstvi"})], ignore_index=True)[PracovniDatabaze.columns]

Data_ORP = AgregaceORP(Nazvy_dict, PuvodniData.copy())
Data_Kraje = AgregaceKraje(PuvodniData.copy())
with pd.ExcelWriter("Statistika_Bilance.xlsx") as writer:
    PuvodniData.to_excel(writer, sheet_name="Vybilancovano_ZUJ", index=False, encoding="utf-8-sig", header=True)
    Data_ORP.to_excel(writer, sheet_name="Vybilancovano_ORP", index=False, encoding="utf-8-sig", header=True)
    Data_Kraje.to_excel(writer, sheet_name="Vybilancovano_Kraje", index=False, encoding="utf-8-sig", header=True)

with pd.ExcelWriter("PDISOH.xlsx") as writer:
    PracovniDatabaze.to_excel(writer, sheet_name="Pracovní databáze", index=False, encoding="utf-8-sig", header=True)
```

Obrázek 17: Implementace funkce „Statistika_Bilance“ – Tvorba výstupních souborů.

6.6 Funkce „SeznamObci“

Tato funkce s využitím importovaných dat týkajících se kódování ZÚJ a jejich zařazení do ORP a krajů umožňuje vytvořit datovou strukturu „dict“, která je dále využívána při agregaci výsledků na větší územní celky a následném formátování výpisu. V souboru „PDISOH.xlsx“ je díky těmto informacím poskytnuta dodatečná informace k jednotlivým záznamům. Příslušná implementace je uvedena na Obr. 18.

```python
def SeznamObci():
    DirPath = str(pathlib.Path().resolve())
    file = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Obce")
    file.columns = file.iloc[0]
    file = file.drop(0)
    file.reindex()
    Kraje_dict = {
        "0": "Hlavní město Praha",
        "1": "Jihočeský",
        "2": "Jihomoravský",
        "3": "Karlovarský",
        "4": "Královehradecký",
        "5": "Liberecký",
        "6": "Moravskoslezský",
        "7": "Olomoucký",
        "8": "Pardubický",
        "9": "Plzeňský",
        "10": "Středočeský",
        "11": "Ústecký",
        "12": "Vysočina",
        "13": "Zlínský"
    }
    Seznam = {}
    for id, i in file.iterrows():
        Seznam[i["Evident"]] = [i["Nazev"], Kraje_dict[str(i["Kraj"])], i["Nalezi_ORP_Kod"]]
    return Seznam
```

Obrázek 18: Implementace funkce „SeznamObci“.

6.7 Agregační funkce

Agregační funkce s využitím slovníku obsahujícím informace o územním členění umožňují vytvořit souhrnné informace na úrovni ORP či krajů. Funkce jsou využívány při formátování výpisů, kde v souboru „Statistika_Bilance.xlsx“ jsou funkce využity při tvorbě jednotlivých listů. Funkce jsou rozděleny dle úrovně agregace a jejich implementace je uvedena na Obr. 19.

```python
def AgregaceORP(Nazvy_dict, Data):
    Data = Data.drop(columns=["Evident - Typ subjektu"])
    for id, i in Data.iterrows():
        Data.loc[id, "Evident"] = Nazvy_dict[int(Data.loc[id, "Evident"])][2]
        Data.loc[id, "Evident - Nazev"] = Nazvy_dict[int(Data.loc[id, "Evident"])][0]
    Data = Data.groupby(by=["Evident", "Evident - Nazev", "Evident - Kraj"]).sum().reset_index()
    return Data


def AgregaceKraje(Data):
    Data = Data.drop(columns=["Evident", "Evident - Typ subjektu", "Evident - Nazev"])
    Data = Data.groupby(["Evident - Kraj"]).sum().reset_index()
    return Data
```

Obrázek 19: Implementace funkcí „AgregaceORP“ a „AgregaceKraje“.

6.8 Funkce definující kódy nakládání a indikátory

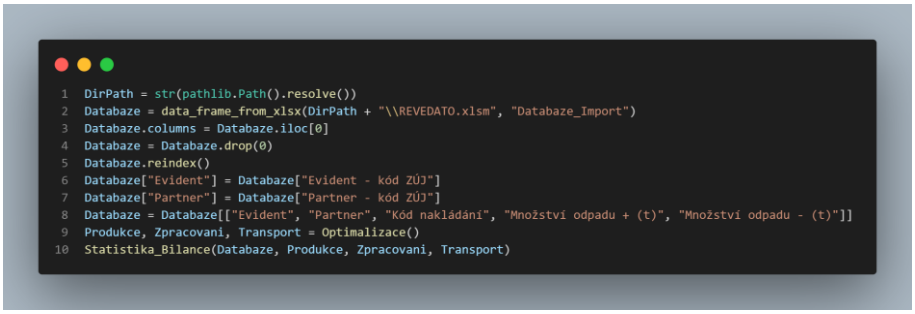
Následující funkce tvoří základ pro správnou identifikaci jednotlivých záznamů dle kódů nakládání a zařazení do příslušného indikátoru odpadového hospodářství. Pro implementaci (viz Obr. 20) kódů je využita struktura „dict“ a formátování textových řetězců do podoby „regular expression“, které nabízí přehledný zápis a efektivní vyhledávání v rámci řetězců.

```
def Fce_Kody_Nakladani_Indikatory():
    Kody_Nakladani_Indikatory = {
        "Produkce": re.compile(r"^(A00|AN30|BN30)|^\w(N60)$"),
        "Vyskladnění": re.compile(r"^(C00)$"),
        "Import": re.compile(r"^B(N6|N16)$"),
        "Produkce ostatní": re.compile(r"^(BN40)|^\w(N50)$"),
        "Materiálově": re.compile(r"^\w(R2|R3|R4|R5|R6|R7|R8|R9|R10|R11|R12|N1|N11|N12|N13|N15)$"),
        "Energeticky": re.compile(r"^\w(R1)$"),
        "Spalování": re.compile(r"^\w(D10)$"),
        "Skládkování": re.compile(r"^\w(D1|D5|D12)$"),
        "Jiné uložení": re.compile(r"^\w(D3|D4)$"),
        "Export": re.compile(r"^\w(N7|N17)$"),
        "Naskladnění": re.compile(r"^\w(N5|R13|D15)$"),
        "Zpracování ostatní": re.compile(r"^\w(D2|D8|D9|D13|D14|N9|N14|N18|N53|N63)$"),
        "Převzetí": re.compile(r"^(B00)$"),
        "Předání": re.compile(r"^\w(N2|N3|N8|N10)$")
    }
    return Kody_Nakladani_Indikatory


def Fce_Kody_Nakladani():
    Kody_Nakladani = {
        "A00": re.compile(r"^(A00)$"),
        "BN30": re.compile(r"^(AN30|BN30)$"),
        "XN60": re.compile(r"^\w(N60)$"),
        "C00": re.compile(r"^(C00)$"),
        "BN6": re.compile(r"^(BN6)$"),
        "BN16": re.compile(r"^(BN16)$"),
        "BN40": re.compile(r"^(BN40)$"),
        "XN50": re.compile(r"^\w(N50)$"),
        "XD1": re.compile(r"^\w(D1)$"),
        "XD2": re.compile(r"^\w(D2)$"),
        "XD3": re.compile(r"^\w(D3)$"),
        "XD4": re.compile(r"^\w(D4)$"),
        "XD5": re.compile(r"^\w(D5)$"),
        "XD8": re.compile(r"^\w(D8)$"),
        "XD9": re.compile(r"^\w(D9)$"),
        "XD10": re.compile(r"^\w(D10)$"),
        "XD12": re.compile(r"^\w(D12)$"),
        "XD13": re.compile(r"^\w(D13)$"),
        "XD14": re.compile(r"^\w(D14)$"),
        "XD15": re.compile(r"^\w(D15)$"),
        "XR1": re.compile(r"^\w(R1)$"),
        "XR2": re.compile(r"^\w(R2)$"),
        "XR3": re.compile(r"^\w(R3)$"),
        "XR4": re.compile(r"^\w(R4)$"),
        "XR5": re.compile(r"^\w(R5)$"),
        "XR6": re.compile(r"^\w(R6)$"),
        "XR7": re.compile(r"^\w(R7)$"),
        "XR8": re.compile(r"^\w(R8)$"),
        "XR9": re.compile(r"^\w(R9)$"),
        "XR10": re.compile(r"^\w(R10)$"),
        "XR11": re.compile(r"^\w(R11)$"),
        "XR12": re.compile(r"^\w(R12)$"),
        "XR13": re.compile(r"^\w(R13)$"),
        "XN1": re.compile(r"^\w(N1)$"),
        "XN5": re.compile(r"^\w(N5)$"),
        "XN7": re.compile(r"^\w(N7)$"),
        "XN9": re.compile(r"^\w(N9)$"),
        "XN11": re.compile(r"^\w(N11)$"),
        "XN12": re.compile(r"^\w(N12)$"),
        "XN13": re.compile(r"^\w(N13)$"),
        "XN14": re.compile(r"^\w(N14)$"),
        "XN15": re.compile(r"^\w(N15)$"),
        "XN17": re.compile(r"^\w(N17)$"),
        "XN18": re.compile(r"^\w(N18)$"),
        "XN53": re.compile(r"^\w(N53)$"),
        "XN63": re.compile(r"^\w(N63)$"),
        "Převzetí": re.compile(r"^(B00)$"),
        "Předání": re.compile(r"^\w(N2|N3|N8|N10)$")
    }
    return Kody_Nakladani
```

Obrázek 20: Sekce Zpracování dat – statistika transportu a kontrola databáze.

6.9 Inicializace výpočetního modulu

Výpočetní modul je inicializován načtením původní databáze a filtrací pouze vybraných sloupců. Následně jsou postupně spouštěny dílčí funkce, které zajišťují samotnou optimalizaci či výpis rekonstruovaných výsledků. Implementovaný kód je uveden na Obr. 21.



```
DirPath = str(pathlib.Path().resolve())
Databaze = data_frame_from_xlsx(DirPath + "\\REVEDATO.xlsm", "Databaze_Import")
Databaze.columns = Databaze.iloc[0]
Databaze = Databaze.drop(0)
Databaze.reindex()
Databaze["Evident"] = Databaze["Evident - kód ZÚJ"]
Databaze["Partner"] = Databaze["Partner - kód ZÚJ"]
Databaze = Databaze[["Evident", "Partner", "Kód nakládání", "Množství odpadu + (t)", "Množství odpadu - (t)"]]
Produkce, Zpracovani, Transport = Optimalizace()
Statistika_Bilance(Databaze, Produkce, Zpracovani, Transport)
```

Obrázek 21: Implementace inicializace výpočetního modulu.

Reference

MŽP, 2022. Matematické vyjádření výpočtu „Soustavy indikátorů OH“ v souladu s vyhláškou č. 273/2021 Sb. (č. 383/2001 Sb.), o podrobnostech nakládání s odpady, v platném znění. Dostupné z: https://www.mzp.cz/C1257458002F0DC7/cz/odpady_podrubrika/$FILE/OODP-Metodika_indikatory_final-20221031.pdf

MŽP, 2021. Vyhláška o Katalogu odpadů a posuzování vlastností odpadů (Katalog odpadů). Dostupné z https://www.mzp.cz/www/platnalegislativa.nsf/26B2B93E9CCDE5B0C125865B002C4914/%24file/VYHL_010521_OL.pdf

CEVOOH, 2022. Souhrnná výzkumná zpráva M1F2 k V1.F.2.1: Zpráva popisující koncept nástroje pro modelování toku odpadů na základě historických dat.