Technická univerzita v Košiciach
Fakulta elektrotechniky a informatiky

Predmet: Heuristické optimalizačné procesy (HOP)

Assignment 2 – Úloha 1
Rozpis služieb záhradkárskej komunity

Študenti:
Antonov Volodymyr
Onishchenko Vladyslav
Morhai Vladyslav
Lysytsia Oleksandr

Študijný program: Inteligentné systémy
Akademický rok: 2025/2026

Košice, 7. decembra 2025

1 Úvod

Cieľom úlohy je zostaviť rozpis služieb pre členov záhradkárskej komunity na obdobie **16 týždňov** (spolu 112 kalendárnych dní) tak, aby bola každý deň zabezpečená ochrana územia v nasledujúcom režime:

- 1 člen na dennej (24-hodinovej) službe,
- 2 členovia na nočnej službe v ten istý kalendárny deň.

Každý člen má:

- preferované dni v týždni (1 = pondelok, ..., 7 = nedeľa),
- prípadné *zákazy* pre konkrétne dni v týždni $E\{d\}$, v ktorých nesmie byť zaradený na službu.

Výsledkom má byť textový súbor so 112 riadkami, kde každý riadok reprezentuje jeden kalendárny deň a obsahuje 3 celé čísla: index člena na dennej službe a indexy dvoch členov na nočnej službe. Indexy členov sú 1-based a zodpovedajú poradiu riadkov vo vstupnom súbore.

Optimalizačná úloha je rozvrhovacia (assignment/scheduling) s pevnými obmedzeniami (*hard constraints*) a viac-kriteriálnym cieľom. Riešenia porovnávame v lexikografickom poradí:

1. platnosť rozpisu (žiadne porušenie pevných obmedzení),
2. spokojnosť preferencií (PrefScore),
3. pokrytie územia (CoverageScore) ako rozlišovací faktor.

V tejto správe najprv rozoberieme formálnu analýzu úlohy, následne navrhnutý model, použitý heuristický algoritmus a napokon experimentálne výsledky implementácie v jazyku Python.

2 Analýza úlohy

Nech:

- $D = \{1, \ldots, 112\}$ – množina kalendárnych dní,
- $M = \{1, \ldots, V\}$ – množina členov podľa vstupného súboru,
- $\text{weekday}(d) \in \{1, \ldots, 7\}$ – deň v týždni pre kalendárny deň d,
- $\text{Prefs}(m) \subseteq \{1, \ldots, 7\}$ – preferované dni v týždni člena m,
- $E(m) \subseteq \{1, \ldots, 7\}$ – zakázané dni v týždni pre člena m.

Pre každý deň $d \in D$ priraďujeme trojicu členov $S_d = (a_d, b_d, c_d)$, kde:

- a_d – člen na dennej službe,
- b_d – člen na prvej nočnej službe,
- c_d – člen na druhej nočnej službe.

2.1 Pevné obmedzenia (Hard Constraints)

Rozpis je platný, ak sú splnené nasledujúce podmienky:

HC1 Zákazy (availability). Pre každý deň d a každého člena $m \in S_d$ platí:

$$\text{weekday}(d) \notin E(m).$$

HC2 Spravodlivosť (fairness) pre denné a nočné služby. Denné služby: každý člen má buď $\lfloor 112/V \rfloor$ alebo $\lceil 112/V \rceil$ denných služieb, pričom súčet denných služieb je $\sum_{m \in M} \text{DayCount}(m) = 112$.

Nočné služby: každý člen má buď $\lfloor 224/V \rfloor$ alebo $\lceil 224/V \rceil$ nočných služieb, pričom súčet nočných služieb je $\sum_{m \in M} \text{NightCount}(m) = 224$.

Maximálny rozdiel medzi počtom denných služieb ľubovoľných dvoch členov je najviac 1 a rovnako pre nočné služby.

HC3 Obmedzenie na deň. Člen nesmie byť v jeden kalendárny deň zaradený viackrát:

$$a_d, b_d, c_d \text{ sú navzájom rôzne pre každý } d \in D.$$

Porušenie akéhokoľvek z týchto obmedzení robí rozpis neplatným.

2.2 Mäkké ciele (Soft Constraints)

Ak sú splnené všetky hard constraints, rozpis porovnávame podľa týchto kritérií:

SC1 Preferencie (PrefScore). Maximalizujeme počet priradení, kde člen slúži v preferovanom dni v týždni.

SC2 Pokrytie územia (CoverageScore). Predpokladáme lineárny model ulice, kde člen i zodpovedá za záhradku s indexom i a súčasne za interval okolo zrkadlovej záhradky $V - i + 1$. Pre každý deň d meriame šírku pokrytia $|Zone(a_d) \cup Zone(b_d) \cup Zone(c_d)|$, normalizovanú hornou hranicou teoreticky dosiahnuteľného pokrytia.

Poradie cieľov je lexikografické:

$$\text{Platnosť} \Rightarrow \text{PrefScore} \Rightarrow \text{CoverageScore}.$$

3 Model riešenia

3.1 Reprezentácia členov

Členovia sú indexovaní $1, \ldots, V$ podľa poradia vo vstupnom súbore. Pre každého člena m uchovávame:

- $\text{Prefs}(m) \subseteq \{1, \ldots, 7\}$,
- $E(m) \subseteq \{1, \ldots, 7\}$.

V implementácii sú tieto informácie uložené v slovníku:

`guards[m]["prefs"]`, `guards[m]["forbiddens"]`,

kde kľúč m je 1-based index člena.

3.2 Model ulice a zón pokrytia

Vstup neobsahuje skutočné priestorové rozloženie záhradiek, preto zavádzame minimalistický model:

- člen i zodpovedá za záhradku i pozdĺž lineárnej ulice,
- záhradky sú indexované $1, \ldots, V$ po jednej strane a späť po druhej,
- zrkadlová funkcia:

$$\text{mirror}(i) = V - i + 1.$$

Zóna jedného člena i je definovaná ako zjednotenie dvoch orezaných intervalov:

$$Zone(i) = [\max(1, i-5), \min(V, i+5)] \ \cup \ [\max(1, \text{mirror}(i)-5), \min(V, \text{mirror}(i)+5)].$$

Pokrytie dňa d je potom:

$$CoverageScore(d) = \frac{\left|Zone(a_d) \cup Zone(b_d) \cup Zone(c_d)\right|}{\min(V, 66)} \in (0, 1].$$

3.3 Reprezentácia rozpisu

Rozpis reprezentujeme ako pole trojíc:

$$S = \{S_d = (a_d, b_d, c_d) \mid d \in D\},$$

kde a_d, b_d, c_d sú indexy členov.

V implementácii používame:

`shifts[d] = [g_day, g_night1, g_night2]`,

pričom indexovanie dní v poli je 0-based (0..111), ale pri zápise výstupu sa indexy členov konvertujú na 1-based.

3.4 Kritériá kvality

PrefScore

Per-deň metrika:

$$PrefScore(d) = \sum_{m \in S_d} \mathbf{1}\big[\text{weekday}(d) \in \text{Prefs}(m)\big].$$

Súhrnná a normalizovaná metrika:

$$TotalPrefScore = \sum_{d \in D} PrefScore(d),$$

$$PrefScore_{norm} = \frac{TotalPrefScore}{3 \cdot 112} \in [0, 1].$$

CoverageScore

Pre každý deň d:

$$CoverageScore(d) = \frac{\left|Zone(a_d) \cup Zone(b_d) \cup Zone(c_d)\right|}{\min(V, 66)}.$$

Súhrnný CoverageScore používame ako priemer za všetky dni:

$$CoverageScore_{avg} = \frac{1}{112} \sum_{d \in D} CoverageScore(d).$$

Fairness penalty

Pre hodnotenie spravodlivosti z pohľadu *celkového* počtu služieb zavádzame pomocnú penalizáciu. Nech:

$$S(m) = \text{DayCount}(m) + \text{NightCount}(m),$$

$$S_{\min} = \min_{m \in M} S(m), \quad S_{\max} = \max_{m \in M} S(m).$$

Definujeme:

$$FairnessPenalty = \max\bigl(0,\ S_{\max} - S_{\min} - 1\bigr).$$

Hodnota 0 znamená ideálny stav (rozdiel najviac 1), kladná hodnota znamená, že niektorí členovia majú o viac než 1 službu navyše oproti najmenej zaťaženým.

4 Návrh algoritmu

Zvolený postup kombinuje:

- konštrukčnú heuristiku (generovanie počiatočného platného rozpisu),
- lokálnu optimalizáciu (2-opt medzi rovnakými slotmi),
- dodatočnú reparáciu spravodlivosti pre nočné služby.

4.1 Konštrukcia počiatočného rozpisu (`init_solution`)

Kvóty

Pre dané V definujeme:

$$q_{day} = \left\lfloor \frac{112}{V} \right\rfloor, \quad r_{day} = 112 \bmod V,$$

$$q_{night} = \left\lfloor \frac{224}{V} \right\rfloor, \quad r_{night} = 224 \bmod V.$$

Každý člen dostane pre denné služby buď q_{day} alebo $q_{day}+1$, pričom presne r_{day} členov dostane $q_{day}+1$. Analogicky pre nočné služby.

Poradie dní podľa náročnosti

Pre každý kalendárny deň d vieme určiť:

- príslušný deň v týždni weekday(d),
- množinu kandidátov – členov, ktorí v daný deň:
  - nemajú zákaz (weekday(d) $\notin E(m)$),
  - ešte neprekročili svoje kvóty pre denné/nočné služby.

Dni triedime podľa rastúceho počtu kandidátov; ako prvé plánujeme tie dni, ktoré majú najmenej povolených kandidátov, čím znižujeme riziko slepých uličiek.

Greedy priraďovanie služieb

Pre každý deň v zoradenom poradí:

1. vyberieme kandidáta na dennú službu:
   - nesmie mať zákaz v daný deň,
   - nesmie mať pridelenú inú službu v tom istom dni,
   - preferujeme:
     (a) člena s preferovaným dňom v týždni,
     (b) člena s menším počtom doterajších denných služieb,
     (c) pri rovnosti člena s menším indexom.
2. analogickým pravidlom priraďujeme oboch členov na nočné služby (dva sloty), pričom sledujeme kvóty pre nočné služby a zakazujeme viac než jednu službu na deň.

Výsledkom je platný rozpis *init_solution*, ktorý spĺňa všetky hard constraints a distribuuje denné aj nočné služby takmer rovnomerne podľa kvót.

4.2 Lokálna optimalizácia (`optimize_schedule`)

Lokálna optimalizácia používa jednoduchý 2-opt krok *medzi rovnakými slotmi*:

- náhodne zvolíme dva rôzne dni $d_1 \neq d_2$,
- náhodne zvolíme slot $s \in \{\text{day}, \text{night1}, \text{night2}\}$,
- v tomto slote vymeníme príslušných členov medzi dňami.

Nový rozpis S' porovnávame s aktuálnym S pomocou funkcie `_is_new_schedule_better()`:

1. Ak S' nie je platný (porušuje niektoré hard constraint), ťah zamietneme.
2. Vypočítame $PrefScore_{norm}(S')$ a porovnáme s $PrefScore_{norm}(S)$:
   - ak rozdiel presahuje malú toleranciu ε, preferujeme vyššiu hodnotu,
   - ak je $PrefScore$ horší o viac ako ε, ťah zamietneme.
3. Pri takmer rovnakom PrefScore porovnávame $CoverageScore_{avg}$:
   - vyšší CoverageScore je lepší,
   - nižší o viac než ε ťah zamietne.
4. Pri rovnosti oboch metrík porovnávame $FairnessPenalty$:
   - nižšia penalizácia je lepšia.

5. Pri úplnej rovnosti všetkých metrík môže byť ťah akceptovaný s malou pravdepodobnosťou (napr. 0.5) ako náhodný tie-break.

Takto iterujeme pevný počet krokov `max_iter`, pričom si vždy ponechávame aktuálne najlepšie riešenie.

4.3 Reparácia fairness pre nočné služby (`repair_night_fairness`)

Po lokálnej optimalizácii získame rozpis $S^{(opt)}$, ktorý má veľmi vysoké PrefScore a CoverageScore, ale nočné služby sa môžu líšiť od ideálneho rozdelenia (q_{night} alebo $q_{night}+1$). Preto aplikujeme dodatočný deterministický *repair*:

1. Vypočítame NightCount(m) pre všetkých členov.
2. Definujeme:

$$Low = \{m \mid \text{NightCount}(m) < q_{night}\}, \quad High = \{m \mid \text{NightCount}(m) > q_{night}\}.$$

3. Pre každého člena $m_{low} \in Low$ sa pokúšame nájsť deň d a nočný slot (night1 alebo night2), kde:
   - v tomto slote slúži niekto z množiny $High$,
   - m_{low} nemá v ten deň inú službu,
   - weekday(d) $\notin E(m_{low})$.
4. Ak taký deň a slot nájdeme, presunieme nočnú službu z $m_{high} \in High$ na m_{low}, aktualizujeme NightCount a pokračujeme.

Táto fáza nemení denné služby, neporušuje zakázané dni ani obmedzenie „max. jedna služba za deň“ a smeruje rozdelenie nočných služieb k ideálnemu stavu q_{night} alebo $q_{night}+1$.

5 Implementácia

5.1 Použitý jazyk a nástroje

Riešenie je implementované v jazyku **Python 3.12** a správa závislostí je riešená cez nástroj **uv**. Projekt je rozdelený do viacerých modulov:

- `main.py` – vstupný bod programu (načítanie vstupu, spustenie algoritmu, zápis výstupu),
- `utils/data_utils.py` – parsovanie `input.txt` a zápis `output.txt`,
- `utils/algo_utils.py` – implementácia `init_solution`, `optimize_schedule`, `repair_night_fai` a pomocných funkcií,
- `utils/helpers.py` – pomocné funkcie (deň v týždni, validácia rozpisu a pod.),
- `utils/print_utils.py` – formátovaný výpis metrík a výsledkov.

5.2 Vstup a výstup

Vstup. Očakáva sa súbor:

`data/input/input.txt`

v tvare definovanom v zadaní: jeden riadok na člena, preferované dni a prípadné zákazy (označené `E<d>`).

Výstup. Výsledný rozpis sa uloží do:

`data/results/output.txt`

kde každý riadok obsahuje tri celé čísla:

$$\langle guard\_day \;\; guard\_night1 \;\; guard\_night2 \rangle,$$

indexy sú 1-based, prvý riadok zodpovedá prvému pondelku, čísla sú oddelené jednou medzerou a riadok končí iba znakom konca riadku.

5.3 Deterministickosť

Na zabezpečenie reprodukovateľnosti výsledkov je fixovaný seed generátora náhodných čísel. Napriek použitiu náhodných 2-opt krokov tak program produkuje deterministický rozpis pre daný vstup.

6 Experimentálne výsledky

Táto časť sumarizuje výsledky behu algoritmu na poskytnutom vstupnom súbore `input.txt`.

6.1 Metodika

Porovnávame tri fázy:

1. **Počiatočný rozpis** (`init_solution`) – výsledok konštrukčnej heuristiky.
2. **Lokálne optimalizovaný rozpis** – po 2-opt lokálnom hľadaní.
3. **Finálny rozpis po repair** – po aplikácii `repair_night_fairness`.

Pre každú fázu meriame:

- normalizovaný PrefScore,
- priemerný CoverageScore,
- FairnessPenalty a hodnoty $S_{\min}, S_{\max}$.

6.2 Výsledky

Príklad nameraných výsledkov:

Fáza	PrefScore_{norm}	$\text{CoverageScore}_{avg}$	FairnessPenalty
Počiatočný rozpis	0.9911	0.6150	2
Po 2-opt optimalizácii	1.0000	0.8977	2
Po `repair_night_fairness`	0.9970	0.8942	1

Pre finálny rozpis platí:

- $S_{\min} = 4$, $S_{\max} = 6$, takže $FairnessPenalty = 1$,
- rozdelenie denných služieb:
  - 34 členov má 1 dennú službu,

  - 39 členov má 2 denné služby,
- rozdelenie nočných služieb:
  - 68 členov má 3 nočné služby,
  - 5 členov má 4 nočné služby.

Čas behu na danom vstupnom súbore je približne 3 sekundy na štandardnom notebooku.

6.3 Diskusia

- Počiatočný rozpis už dosahuje veľmi vysoké PrefScore vďaka preferenčne orientovanej konštrukčnej heuristike.
- Lokálna optimalizácia (2-opt) ďalej zvyšuje CoverageScore, pričom PrefScore zostáva na úrovni 1.0.
- Fáza `repair_night_fairness` mierne znižuje PrefScore a CoverageScore, ale zreteľne zlepšuje spravodlivosť rozdelenia nočných služieb (znižuje rozdiel v total počtoch služieb).
- Všetky fázy zachovávajú platnosť rozpisu (žiadne porušenie zákazov, max. jedna služba za deň).

7 Záver

V tejto práci sme vyriešili úlohu rozpisu služieb pre záhradkársku komunitu na obdobie 16 týždňov s pevnými obmedzeniami na dostupnosť členov, spravodlivé rozdelenie denných a nočných služieb a obmedzením na maximálne jednu službu za deň.

Navrhli a implementovali sme model, ktorý:

- reprezentuje preferencie a zákazy ako množiny dní v týždni,
- zavádza jednoduchý lineárny model ulice a zón pokrytia,
- definuje kvóty pre denné a nočné služby s rozdielom najviac 1.

Použitý algoritmus kombinuje konštrukčnú heuristiku, lokálnu 2-opt optimalizáciu a dodatočnú reparáciu fairness pre nočné služby. Experimentálne výsledky ukazujú, že:

- dosahujeme veľmi vysokú spokojnosť preferencií ($\text{PrefScore}_{norm} \approx 0{,}997$),
- výrazne zvyšujeme pokrytie územia oproti počiatočnému riešeniu,
- udržiavame rozumnú spravodlivosť rozdelenia služieb (FairnessPenalty = 1).

Možnými smermi ďalšieho zlepšenia sú:

- implementácia 1-opt krokov v rámci dňa,
- použitie metaheuristík (napr. simulated annealing) pre rozsiahlejšie prehľadávanie priestoru riešení,
- jemnejšie váhovanie medzi PrefScore a CoverageScore podľa preferencií zadávateľa.

Napriek tomu aktuálne riešenie spĺňa požiadavky zadania, generuje platný rozpis a dosahuje dobrý kompromis medzi preferenciami, pokrytím a spravodlivým rozdelením záťaže medzi členov.