Diskrétní simulace za použití knihovny SimPy

Panovský Tomáš

5. listopadu 2025

1 Úvod

V této bakalářské práci se zabývám diskrétní simulací za použití knihovny SimPy v Pythonu. Knihovna Simpy umožňuje modelovat procesy, jenž probíhají souběžně, a mohou být zastaveny, nebo pozastaveny na určitou dobu. Praktická část bakalářské práce obsahuje aplikaci simulující průběh hudebního festivalu. Cílem simulace je zjistit, jak se návštěvníci pohybují a kde vznikají fronty, což může pomoci při organizaci reálného festivalu.

2 Systém

str17 Reálnou skutečnost, kterou chceme analyzovat, označujeme jako systém. Analyzovat znamená sledovat, jak se různé části systému chovají v čase, a vyvozovat závěry o efektivitě, kapacitě nebo chování systému.

Na systémy můžeme nahlížet jako diskrétní nebo spojité, přičemž záleží na úhlu pohledu a vlastnostech, které v systému převažují. Diskrétní systém je takový, u kterého se stavové proměnné mění pouze v diskrétních časových okamžicích. Příkladem diskrétního systému je hudební festival, kde se stavové proměnné, například počet návštěvníků ve frontě u stánku s pivem, mění pouze tehdy, když návštěvník přijde na řadu, nebo dostane pivo a odejde. Oproti tomu spojitý systém je takový systém, u kterého se stavové proměnné mění plynule v čase, nikoli pouze v diskrétních okamžicích.

Stav systému je definován jako soubor stavových proměnných, které jsou nezbytné k popisu systému v libovolném okamžiku. V simulaci hudebního festivalu mohou být možné stavové proměnné například počet lidí čekajících ve frontách u stánků s občerstvením, počet lidí právě sledujících koncert, nebo čas příchodu dalšího návštěvníka do areálu.

2.1 Komponenty systému

Každý systém lze chápat jako soubor vzájemně propojených prvků, které společně ovlivňují jeho chování. Aby bylo možné systém lépe pochopit, je vhodné rozdělit jej na základní komponenty, které popisují jeho strukturu a dynamiku. Mezi tyto komponenty patří entity, jejich atributy, aktivity a události.

Entita je objekt zájmu v systému. V našem případě mohou být entitami například návštěvníci festivalu, stánky s občerstvením nebo pódium.

Atribut je vlastnost entity, například počet lidí u stánku, nebo informace o tom, zda má návštěvník hlad nebo je unavený.

Aktivita představuje časově omezenou činnost entity, například čekání ve frontě, sledování koncertu nebo nákup jídla.

Událost je okamžitá změna stavu systému. Události dělíme na Endogenní a Exogenní. Endogenní probíhají uvnitř systému a jsou způsobeny chováním jeho komponent, např. návštěvník dokončí konzumaci jídla a odchází od stánku. Exogenní probíhají v prostředí systému a ovlivňují ho zvenčí, například náhlý déšť.

3 Úvod do simulace

str6 Simulace je napodobení systému v čase. Cílem je vytvořit umělou historii stavu daného systému, a následně pozorovat tuto uměle vytvořenou historii za účelem vyvození závěrů o provozních vlastnostech systému, tedy o měřitelných charakteristikách a výkonnosti systému, jako je například délka front u stánků, průměrná doba čekání návštěvníků nebo hustota návštěvníků u pódia.

Chování systému vyvíjejícího se v čase se zkoumá pomocí simulačního modelu. V této práci je simulační model vytvořen v knihovně SimPy. Simulační model poté může být použit k prozkoumání široké škály otázek typu „co by se stalo, kdyby“ týkajících se reálného systému. Například můžeme zkoumat, zda je daný počet stánků s občerstvením dostatečný pro obsloužení všech návštěvníků festivalu bez dlouhých front.

Možné změny systému pak lze nejprve nasimulovat, aby bylo možné předpovědět jejich dopad na výkonnost systému. Simulace může být také použita ke studiu systémů ve fázi návrhu, ještě před jejich samotnou realizací.

4 Model systému

str. 13 **Model je definován jako reprezentace systému za účelem jeho studia.** Pro většinu studií je nutné zvažovat pouze ty aspekty systému, které ovlivňují problém, který je předmětem zkoumání.

Modely lze klasifikovat jako statické nebo dynamické, deterministické nebo stochastické a diskrétní nebo spojité. Statický simulační model reprezentuje systém v určitém časovém okamžiku, zatímco dynamické modely reprezentují systémy, jak se mění v čase. Deterministické simulační modely neobsahují žádné náhodné prvky, a to ani ve vstupních datech, ani v průběhu samotné simulace. Při opakovaném spuštění se stejnými vstupy poskytují vždy totožný průběh i výsledek simulace. Stochastické modely naproti tomu obsahují jeden nebo více náhodných prvků, které mohou vstupovat do simulace jak na jejím začátku, tak v jejím průběhu, například při generování časů událostí nebo rozhodování o chování entit. Díky tomu lépe vystihují systémy, jejichž chování je ovlivněno náhodou. Výsledky takových simulací nejsou jednoznačné, ale mají pravděpodobnostní charakter. Diskrétní a spojité modely jsou definovány obdobně jako u systémů.

Pro studium hudebního festivalu použijeme diskrétní, dynamický a stochastický model. Diskrétní model, protože stavové proměnné se mění pouze v konkrétních okamžicích. Dynamický model, protože sleduje vývoj systému v čase během celé doby trvání festivalu. Stochastický model, protože některé vstupy, například časy příchodů návštěvníků, doba čekání u stánku nebo délka sledování koncertu, jsou náhodné.

5 Simulace v SimPy

Pro realizaci diskrétní simulace jsem zvolil jazyk Python a knihovnu SimPy. Ta je postavena především na generátorech využívající příkaz `yield`, které umožňují popis chování entit prostřednictvím jejich aktivit a událostí.

5.1 Generátory v Pythonu

Na hodnoty, které se skládají z lineárně uspořádaných prvků, můžeme pohlížet jako na posloupnosti. Například pole [1, 2, 3] můžeme chápat jako posloupnost hodnot 1, 2 a 3. Posloupnosti se také nazývají iterovatelné hodnoty. Průchod posloupností reprezentujeme hodnotou, která se nazývá iterátor.

Generátor je speciální typ funkce, která obsahuje v těle příkaz `yield`. Příkaz `yield` umožňuje generátoru postupně produkovat jednotlivé prvky posloupnosti.

Posloupnost zde znamená řadu hodnot, které generátor postupně poskytuje. Každý prvek posloupnosti se vyhodnotí až v okamžiku, kdy je vyžádán. Příkaz produkující prvek posloupnosti nabývá tvaru: `yield element`. Po provedení příkazu yield se generátor pozastaví a vrátí hodnotu specifikovanou příkazem `yield`. Při dalším volání pokračuje ve vykonávání od místa, kde byl přerušen, dokud není generátor ukončen.

Příklad generátoru:

```
def get_numbers():
   i = 0
   while i < 3:
      yield i
      i = i + 1
```

Napřed pouze vytvoříme iterátor i:

```
i = get_numbers()
```

Další prvek můžeme vyžádat funkcí next. Dalším prvkem posloupnosti bude hodnota určená příkazem yield. Tedy:

```
next(i)
```

V tuto chvíli bude v proměnné i uložena 0. Tělo generátoru se začne vykonávat od pozastaveného místa až po příkaz yield. Vykonávání těla generátoru je pozastaveno na řádku:

```
i = i + 1
```

Popsaným způsobem získáme další prvky posloupnosti, tedy při dalším volání `next(i)` bude v proměnné i 1, a nakonec 2. Skončení vykonávání těla generátoru povede k vyvolání výjimky `StopIteration`

Na rozdíl od běžné funkce, která po provedení svého kódu vrací nějaký element příkazem`return element`, smí generátor příkazem return vrátit pouze hodnotu `None`.

V kontextu diskrétní simulace v knihovně SimPy jsou generátory využity k modelování aktivit, které představují chování jednotlivých entit (např. návštěvníků festivalu). Každý yield odpovídá určité události. Příkladem může být čekání ve frontě u stánku s jídlem, dokončení přípravy jídla, a obdržení jídla zákazníkem s

jeho následnou konzumací. Yield umožňuje přerušit a znovu obnovit průběh aktivity v čase. Takto lze simulovat souběžné aktivity více entit a stochastické prvky, například náhodné časy příprav nebo příchodů návštěvníků, což odpovídá reálnému chování systému.

Příklad využití generátoru ve funkci, která simuluje událost v návštěvníkově aktivitě go_for_food() jenž simuluje návštěvníka, který si jde koupit jídlo:

```
preparation_time = random.randint(time_min, time_max)
yield self.festival.timeout(preparation_time)
print(f"{self.name} {self.surname} dostává {food}")
```

Proměnná preparation_time zde obsahuje vygenerovaný čas přípravy jídla. Aktivita se poté pomocí yield self.festival.timeout(preparation_time) na zadaný čas pozastaví, ale to a další podstatné mechanismy v SimPy vysvětlíme později.

5.2 Základní principy SimPy

Pokud SimPy rozložíme na základní principy, je to jen asynchronní dispečer událostí. Simpy generuje události a plánuje je na konkrétní čas simulace. Události jsou řazeny podle priority, simulačního času a rostoucího ID události. Každá událost má také seznam callback funkcí, které se spustí, když je událost vyvolána a zpracována smyčkou událostí. Události mohou mít také návratovou hodnotu.

Komponenty zapojené do tohoto procesu jsou Environment (prostředí), události a procesní funkce, které napíše programátor vytvářející simulaci. Procesní funkce, tedy python generátorové funkce, které yieldují instance Event, implementují simulační model a tedy definují chování simulace. Environment ukládá tyto události do svého seznamu událostí a sleduje aktuální čas simulace.

Pokud procesní funkce yieldne událost, SimPy přidá proces do callbacků této události a pozastaví jeho vykonávání, dokud není událost vyvolána a zpracována. Když se proces, který čekal na událost, obnoví, obdrží také hodnotu události.

6 Příklad metody

Pro ilustraci uvádím metodu `go_to_festival_area()`, která simuluje průchod návštěvníka vstupním turniketem do areálu festivalu.

```
def go_to_festival_area(self, entrances):

    yield self.festival.timeout(random.expovariate(1/5))
    entrance_id = occupied.index(min(occupied))
    entrance = entrances[entrance_id]
    occupied[entrance_id] += 1

    with entrance.request() as req:
        queue_start = self.festival.now
        yield req

        queue_waiting_time = self.festival.now - queue_start
        entry_time = random.uniform(1, 3)
        yield self.festival.timeout(entry_time)

    self.state["location"] = resources.Location.FESTIVAL_AREA
    occupied[entrance_id] -= 1
```

Metoda `go_to_festival_area()` simuluje vstup návštěvníka jedním z dostupných vstupů. Metoda přijímá dva argumenty, a to instanci návštěvníka - self, a SimPy resource entrances. SimPy resource jsou objekty, ke kterým návštěvníci v simulaci přistupují, entrances je tedy seznam vstupů, mezi kterými si návštěvník může vybrat.

Každý návštěvník dorazí po náhodném zpoždění, které je generováno pomocí `random.expovariate(1/5)`, což modeluje příchody návštěvníků s průměrem 5 časových jednotek. Metoda využívá k simulování zastavení času pro návštěvníka funkci `yield self.festival.timeout()`, která na počet zadaných časových jednotek danou instanci uspí.

Návštěvník si vybere vstup, u kterého je aktuálně nejméně lidí čekajících ve frontě, což je sledováno pomocí globálního seznamu `occupied`. Následně požádá o přístup ke vstupu pomocí `entrance.request()`. Proměnná `queue_waiting_time` uchovává dobu, po kterou návštěvník čekal ve frontě, než se dostal „na řadu“. Po získání přístupu stráví návštěvník náhodnou dobu kontolou během vstupu, simulovanou pomocí `random.uniform(1, 3)` a `yield`

`self.festival.timeout(entry_time)`. Nakonec se počet lidí u daného vstupu sníží, což znamená, že návštěvník dokončil průchod vstupem, a nastaví se mu atribut „location“ na FESTIVAL_AREA.

Tento přístup umožňuje sledovat délku front a čekací doby u jednotlivých vstupů, což poskytuje užitečné informace pro analýzu průchodnosti a vytíženosti vstupů během festivalu.