Diskrétní simulace za použití knihovny SimPy

Panovský Tomáš

28. října 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 V rámci této bakalářské práce budeme reálnou skutečnost, kterou chceme analyzovat, označovat jako systém. V kontextu simulace 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. 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.

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.

2.1 Komponenty systému

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, například čekání ve frontě, sledování koncertu nebo nákup jídla.

Událost (event) je okamžitá změna, která může změnit stav systému. Události mohou být:

- **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 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 Diskrétní simulace

str19 Diskrétní simulace systémů je založená na událostech, což znamená modelování takových systémů, ve kterých se stavové proměnné mění pouze v určitých,

diskrétních časových okamžicích – tedy v momentech, kdy nastane konkrétní událost. Typickým příkladem může být příchod návštěvníka na festival nebo jeho odbavení u vstupu. Mezi těmito událostmi zůstává systém beze změny. Na rozdíl od analytických metod, které využívají matematické rovnice k nalezení řešení, jsou diskrétní simulační modely obvykle řešeny numerickými metodami – tedy „spuštěním“ modelu na počítači. Simulace generuje umělou historii systému na základě předpokládaných pravidel a vstupů.

5 Simulace v SimPy

Simpy je Python knihovna pro diskrétní simulaci událostí a procesů. Procesy jsou v SimPy definovány pomocí Pythonových generátorových funkcí, ty lze využít pro simulaci aktivních entit, jako jsou například zákazníci, vozidla nebo třeba zvířata. SimPy také poskytuje různé typy sdílených zdrojů pro modelování bodů s omezenou kapacitou. Simulace lze provádět „co nejrychleji“, v reálném čase, nebo ručně krokovat jednotlivé 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.