# AUTOMATICKÁ SEKAČKA
# VÝZKUMNÁ ZPRÁVA

**PŘEDMĚT**: 4IT495 Simulace systémů
**AUTOR**: Ladislav Dyntar
**TYP MODELU**: Multiagentní
**PROVEDENO V**: NetLogo [1]

## 1 Definice problému

V dnešní moderní době je snahou valné většiny lidí a organizací ušetřit si co nejvíce práce a/nebo nákladů. Vlastnictví rozlehlého pozemku s sebou, kromě mnoha radostí, přináší i řadu povinností. Jednou z nich je údržba travnatého porostu.

Cílem tohoto modelu je simulovat pohyb automatické sekačky (či sekaček) po rozsáhlé travnaté ploše, na jejíž údržbě chce majitel ušetřit náklady, aniž by při tom musel jakkoliv omezovat využívání daného prostoru. Příkladem takového prostoru může být městský park či hotelová zahrada.

Každá automatická sekačka bude po nastartování automaticky jezdit po vyhrazeném prostoru a sekat trávu. Při své práci se bude vyhybat překážkám, a to jak statickým (stromy, zahradní domek) tak dynamickým (lidé a zvířata) - ty se budou po mapě náhodně pohybovat.

Aby to nebylo tak jednoduché, ne vždy je účinnost sekačky 100%. A tak se může stát, že bude nutné přejet přes určité místo i vícekrát, aby bylo posekáno dle představ majitele trávníku. Pokud však naopak sekačka opakovaně přejede přes daný kus trávy a ten bude již zcela posekaný, bude se daný kus trávy opotřebovávat - tráva se poškodí.

Cílem simulace bude pozorovat vliv jednotlivých algoritmů (a jejich nastavení) na dobu trvání sečení a s tím související růst nákladů. Dále budeme pozorovat četnost poškození trávníku vlivem zvoleného algoritmu. Přitom předpokládáme, že cílem majitele pozemku je posekat zcela celý pozemek a dosáhnout přitom co nejnižšího poškození při co nejmenších nákladech (benzín/čas).

# 2 Metoda

Pro řešení problému formou multiagentního modelu jsem se rozhodl z několika důvodů. V zadání jsem hned na začátku identifikoval několik agentů, jejichž základní vlastností je reaktivní chování - dokáží reagovat na danou situaci a na stav okolních agentů. Každý agent je však nezávislý na těch ostatních, má svůj vlastní cíl. Mým cílem bylo vytvořit i několik základních algoritmů pohybu sekačky, které by reagovaly nejen na již zmiňovaný stav okolních agentů, ale zahrnovaly v sobě i prvky vlastního (náhodného) rozhodování. Systém se tím pádem stal natolik komplexním, že bez návrhu multiagentního systému by bylo téměř nemožné jej pozorovat a analyzovat krok po kroku.

Možnost živé simulace jsem předem zavrhl, a to z několika důvodů. Prvním byl výběr vhodných prostor; druhým časové omezení a s ním související možnost simulovat celý model opakovaně, navíc pokaždé s jinými atributy; a posledním pak technologické omezení, jelikož nedisponuji technikou ani znalostmi vhodnými k sestavení mnou navrhované automatické sekačky (jen pro upřesnění dodávám, že se mi, k mému překvapení, při hledání podařilo najít několik výrobců, kteří se podobnými sekačkami již zabývají).

Rozhodl jsem se tedy využít programu NetLogo verze 5.0, se kterým jsem se seznámil na hodinách předmětu 4IT495 Simulace systémů, vyučovaném na VŠE v Praze.

NetLogo, díky svému jednoduchému, avšak plně upravitelnému grafickému rozhraní umožňuje uživateli navrhovaný systém sledovat přímo za běhu. Je tedy možné ověřit si důsledky jednotlivých nastavení už v rámci samotné simulace a není nutné analyzovat průběh simulace až po jejím ukončení.

Zároveň je třeba zmínit, že problém sekačky v této práci představuje hledání vhodného algoritmu na prohledávání stavového prostoru. Tuto NP-úplnou optimalizační ulohu mnoho lidí zná spíše pod pojmem Problém Obchodního Cestujícího (Traveling Salesman Problem - TSP). Jedná se o velice komplikovaný algoritmus, prakticky neřešitelný optimálně v polynomiálním čase.

# 3 Detailní popis modelu

## 3.1 Popis prostředí

Simulované prostředí představuje čtverec o rozměrech 65x65 patches, složený z menších čtverců o rozměrech 1x1. Pro účely naší simulace předpokládáme, že jedna délka představuje 1 metr v reálném světě.

Simulovaný svět je ze všech stran uzavřený, představme si to jako plot okolo pozemku. Tato hranice není vidět.

Každý čtverec je po úvodní inicializaci tmavě zelený. Tato barva představuje vysoký travní porost, který má automatická sekačka za úkol zkrátit. Barva takové neposekané dlaždice je určena jako "green". Jakmile dojde k posekání vybraného

čtverce, přičemž pravděpodobnost, že bude čtverec posekán je označena jako "pst-posekani-spravne" je jeho barva nastavena na "green + 1".

Pokud přes již posekaný čtverec, označený jako "green + 1", sekačka přejede opět, s pradvěpodobností "pst-posekani-spatne1" dojde k lehkému poničení trávy vlivem opětovného přejetí sekačky a takový čtverec je následně označen barvou "green + 2".

Pokud přes již posekaný čtverec, označený jako "green + 2", sekačka přejede opět, s pradvěpodobností "pst-posekani-spatne2" dojde k střednímu poničení trávy vlivem opětovného přejetí sekačky a takový čtverec je následně označen barvou "green + 3".

Pokud přes již posekaný čtverec, označený jako "green + 3", sekačka přejede opět, s pradvěpodobností "pst-posekani-spatne3" dojde k závažnému poničení trávy vlivem opětovného přejetí sekačky a takový čtverec je následně označen barvou "green + 4".

Pokud přes již posekaný čtverec, označený jako "green + 4", sekačka přejede opět, s pradvěpodobností "pst-posekani-zniceno" dojde k naprostému zničení trávy vlivem opětovného přejetí sekačky a takový čtverec je následně označen barvou "yellow".

V případě, že je v rámci grafického rozhraní zapnuta možnost "auto-rotor-onoff", tedy automatické vypínání rotoru sekačky v moment, kdy přejíždí již posekanou dlaždici, nedochází k poničení trávy vůbec.

Na ohraničeném pozemku se ještě nachází řada překážek. Na mapu jsou náhodně umístěny stromy, které zabírají přesně 1 čtverec. Jejich barva je nastavena jako "brown" a v grafickém rozhraní je možné určit si, kolik stromů chceme vygenerovat.

Další překážkou je zahradní domek, který je na mapu umístěn pevně a představuje objekt o rozměrech 2*4 dlaždice. Jeho barva je taktéž nastavena na "brown".

## 3.2 Agenti

### 3.2.1 Lidé

Jsou generováni v počtu stanoveném uživatelem v grafickém rozhraní. Jejich rychlost pohybu je stanovena náhodně s normálním rozložením se střední hodnotou 2 a směrodatnou odchylkou 0.2. Pro přehlednost byli všichni lidé označení červenou barvou "red". Jejich umístění na mapě je generováno náhodně.

### 3.2.2 Zvířata

Zvířata jsou také generována na základě vstupu uživatele z grafického rozhraní. Na mapě se zobrazují s ikonou psa. Jejich rychlost pohybu po pozemku je

generována náhodně s normálním rozdělením se střední hodnotou 1 a směrodatnou odchylkou 0.2. Jejich barva byla nastavena na "blue", tedy modrou. Jejich umístění na mapě je generováno náhodně.

### 3.2.3 Sekačky

Dalším a pravděpodobně nejdůležitějším agentem jsou samotné sekačky. V grafickém rozhraní lze zvolit jejich konkrétní počet. Pro celou skupinu sekaček máme možnost zvolit, zda se na mapě vygeneruje náhodně, v levém dolním rohu či na středu pozemku. Rychlost sekaček byla zvolena jako 1. Sekačkám je, kromě atributu rychlost, přiřazen ještě atribut vyjadřující jejich algoritmus pohybu po mapě.

Pokud jsme si stanovili jednotku délky jako metry a víme, že délka hrany jedné dlaždice je 1 jednotka a sekačka se pohybuje rychlostí jedné jednotky za tick, můžeme říci, že sekačka ujede 1 metr/tick, přitom jednotkou času může být libovolná dosazená jednotka dle problému, který zrovna model bude řešit. Stejně tak je možné zaměnit metry za jinou jednotku délky dle požadavků řešitele.

## 3.3 Chování agentů

V každém ticku modelu jsou postupně volány metody "*move-lidi*", "*move-zvirata*" a "*move-sekacky*". Lidé i zvířata se pohybují na základě stejného algoritmu - pokud mohou jít rovně vpřed (nejsou na kraji pozemku, před nimi není žádná překážka ani jiný agent) jdou o jim vlastní rychlost vpřed. Pokud narazí, změní směr o "random 360" stupňů a pokračují opět rovně, dokud znovu nenarazí.

Zajímavé to začíná být až u pohybu samotné sekačky. Té v první fázi v grafickém rozhraní přidělíme jeden z navržených algoritmů, případně můžeme sáhnout po možnosti přidělit každému typu sekačky jeden z náhodných algoritmů ze seznamu algoritmů.

Sekaček se týkají i další grafické prvky v rozhraní - slider "*uhel-otoceni*" nám umožňuje zvolit úhel, o který se sekačka bude otáčet v případě, že narazí (implementace se liší v závislosti na zvoleném algoritmu).

Input box "*zmena-polomeru-rozdeleni*" je využit u kruhových algoritmů. Čím vyšší hodnota, tím rychleji se zvětšuje poloměr opisovaného kruhu sekačky.

Sekačky, stejně jako lidé i zvířata, v případě, že mohou pokračovat bez omezení ve svém směru (ať už je přímý či kruhový), v něm pokračují až do té doby, dokud jim to okolí umožňuje. Jakmile se před nimi ocitne překážka, provedou kroky definované ve svém algoritmu a pokračují dále v klasickém pohybu.

Sekačky při svém pohybu navíc ovlivňují patches, přes které zrovna přejíždí - seká na nich imaginární trávu. Pravidla pro sekání trávy sekačkou byla popsána výše v sekci Popis prostředí.

Sekačka eviduje spotřebu benzínů/energie, na každý jeden tick spotřebuje jednu jednotku. Převod jednotek na reálné hodnoty je díky tomu možné uskutečnit až

dodatečně s ohledem na to, bude-li uživatel modelu pracovat s cenou elektřiny, benzínu či jiných zdrojů.

## 3.4 Popis algoritmů sekačky

Aktuálně je v kódu naprogramováno 7 algoritmů, není však nejmenší problém přidat v budoucnu další v případě zájmu o rozvoj modelu.

### 3.4.1 Algoritmus 1: uhel-otoceni

Tento velice jednoduchý algoritmus nejprve kontroluje, zda může pokračovat v cestě (tím budeme i dále v textu rozumět situaci, kdy sekačce v cestě nestojí žádná překážka, lidé, zvířata či se nenachází na kraji pozemku). Pokud je vše v pořádku, jede dopředu o rychlost sekačky "*speed*". Pokud narazí, otočí se doleva o úhel "*uhel-otoceni*" stanovený v grafickém prostředí.

### 3.4.2 Algoritmus 2: random-uhel-otoceni

Tento algoritmus je pouhým rozšířením prvního algoritmu. V případě, že narazí, otočí se sekačka o náhodný úhel "*random uhel-otoceni*", přičemž náhodně vybírá na základě vstupu uživatele z grafického rozhraní.

### 3.4.3 Algoritmus 3: random-360

Tento algoritmus je pouhým rozšířením prvního algoritmu. V případě, že narazí, otočí se sekačka o náhodný úhel "*random 360*".

### 3.4.4 Algoritmus 4: kruh

Algoritmus nazvaný kruh představuje již komplikovanější řešení problému pohybu sekačky. Po úvodní kontrole na možnost pohybu vpřed se, pokud je pohyb umožněn, sekačka posune vpřed a mírně se natočí směrem vlevo. Toto se stále opakuje, takže sekačka by se bez zásahu pohybovala stále v kruhu. Proto s postupem času snižuji úhel, o jaký se sekačka otáčí vlevo, aby se poloměr kruhu pomalu zvětšoval a sekačka tak mohla pokrýt co největší prostor.

Jakmile při svém pohybu sekačka narazí na překážku, otočí se doleva o "*180 - random uhel-otoceni*" stupňů, na chvíli přeruší kroužení a odjede od překážky. Po 25 krocích se opět vrátí ke kroužení, začíná však opět s maximálním úhlem, který byl implicitně zvolen 45 stupňů.

### 3.4.5 Algoritmus 5: kruh s random dojezdem

Tento algoritmus představuje mírné upravení předchozího algoritmu "*kruh*". Vychází z předpokladu, že na začátku je pro sekačku výhodné kroužit v kruzích, ale v pozdější fázi je lepší vyhledávat neposekaná políčka náhodně. Proto se po několika nárazech změní algoritmus sekačky na "*lt 180 - random uhel-otoceni*".

### 3.4.6 Algoritmus 6: dlazdice

Předposlední algoritmus je nazvaný "dlaždice". Jeho náplní je totiž vyhledávat zatím neposekané patche a upřednostňovat je před ostatními, aby bylo co nejvíce minimalizováno opotřebení již posekaných částí pozemku. Algoritmus tedy již posekané dlaždice bere jako svou překážku a snaží se jim vyhybat (pokud je to možné).

Po úvodní kontrole na možnosti pohybu vpřed, pakliže je pohyb umožněn, se sekačka podívá, zda-li je dlaždice před ní neposekaná. Pokud je, přejde na ni. Pokud není, podívá se o "*uhel-otocky-kontroly*" vlevo. Pokud zde najde neposekaný patch, otočí se a čeká na další "tick", aby mohla provést cestu vpřed. Pokud však ani po tomto otočení nenalezne vhodný patch, opakuje hledání v rozmezí 360 stupňů. Pokud ani jeden z okolních patchů sekačky není volný, uskuteční sekačka krok vpřed na již posekaný trávník a zde opakuje hledání o 360 stupňů. Jakmile by zde našla neposekanou část, přesune se na ni. Pokud ji nenajde, opět přejde na již posekané místo a toto opakuje stále dokola.

### 3.4.7 Algoritmus 7: search

Nejpokročilejším algoritmem je pohyb sekačky nazvaný jako "search". Spočívá v tom, že sekačka se rozhlíží kolem sebe v kruhu 360 stupňů nejprve na vzdálenost 1 (viditelnost určuje uživatel v grafickém rozhraní pomocí číselného vstupu "viditelnost-sekacky") a tuto vzdálenost sekačka postupně navyšuje až do hodnoty stanovené v parametru "viditelnost-sekacky". Tento "radius" hledání postupně rozšiřuje až do té doby, dokud ve svém okolí nenalezne neposekanou dlaždici. V případě, že se takovou podaří nalézt, přesune se na ni a opakuje hledání (opět s nulovým úhlem a radiusem 1).

Musíme mít na paměti, že tento algoritmus je řešením pohybu sekačky po námi simulované ploše, v reálném světě by bylo třeba sekačku vybavit výkonnými sensory a pravděpodobně by se její pohyb odvíjel od obrazu zaznamenaného prostředí v paměti sekačky. Neposekaným místem, na které se sekačka musí vracet, by mohl být např. prostor, kde se při původním průjezdu nacházel člověk či zvíře - těm se sekačka samozřejmě musela vyhnout.

## 3.5 Spuštění simulace

Prvotní nastavení modelu provedeme stisknutím tlačítka "*setup*". To spustí inicializaci všech proměných, nastavení mapy, vykreslí sekačky, lidi a zvířata. Spuštění samotné simulace je možné provést stisknutím tlačítka "*go*". Simulace se sama ukončí v moment, kdy jsou všechna políčka na mapě posekaná - takový stav je cílem simulace.

## 3.6 Další omezení a předpoklady

Pro účely naší simulace byly stanoveny následující pravděpodobností hodnoty, určující pravděpodobnost, že dojde k poškození trávníku vlivem opakovaného pohybu sekačky na jedné a té samé dlaždici:

*set pst-posekani-spravne 90*
*set pst-posekani-spatne1 4*

*set pst-posekani-spatne2 3*
*set pst-posekani-spatne3 2*
*set pst-posekani-zniceno 1*

Jak je vidět, jsme poměrně přísní. V reálném světě by se všechny tyto hodnoty blížili spíše k jednomu procentu a méně, pro účely naší simulace však záměrně ponecháme tato kritéria, abychom mohli lépe pozorovat nízkou efektivitu jednotlivých algoritmů. Naopak pravděpodobnost, že sekačka poseče daný patch správně by v reálu mohla dosahovat spíše vyšších hodnot, pro naše účely budeme předpokládat, že se jedná o novou technologii, prozatím s nízkou efektivitou 90 %.

Počet stromů pro všechny naše simulace jsme stanovili na 8; kůlnu vykreslíme vždy. Startovní pozice pro všechny simulace byla uprostřed. Sekačku jsme na trávník vpustili vždy jen jednu protože nepředpokládáme, že by si vlastník zahrady kupoval sekačky dvě.

Proměnná "*škody*" používaná v tabulkách níže znázorňuje součet všech jakkoliv poškozených dlaždic s trávou (*posekano-spatne1-3 + posekano-zniceno*).

Je nutné dále upozornit na to, že v modelu abstrahujeme od prvků jako je vliv počasí, nutnost doplňovat pohonné hmoty, poruchovost sekaček a jiné.

# 4 Výsledky

Simulovali jsme opakovaně několikrát pro každý algoritmus. Simulace navíc proběhla ve dvou variantách - s 10 lidmi a 10 zvířaty na pozemku (označeno jako varianta B) a poté bez zvířat a bez lidí (označeno jako varianta A). Průměrné výsledky a specifika jednotlivých algoritmů si probereme níže.

| A | uhel-otoceni | | random-uhel-otoceni | | random-360 | | dlazdice | |
|---|---|---|---|---|---|---|---|---|
| | ticks | škody | ticks | škody | ticks | škody | ticks | škody |
| 45 | nelze | | nelze | | 61 671,2 | 1 777,2 | 71 543,8 | 1 834,6 |
| 67 | 261 776,4 | 3 340,6 | 91 921,8 | 2 095,8 | 61 671,2 | 1 777,2 | 71 543,8 | 1 834,6 |
| 107 | 61 054,2 | 1 781,4 | 50 671,0 | 1 492,0 | 61 671,2 | 1 777,2 | 71 543,8 | 1 834,6 |
| 159 | 62 151,6 | 1 816,6 | 46 472,0 | 1 407,0 | 61 671,2 | 1 777,2 | 71 543,8 | 1 834,6 |
| avg | 128 327,4 | 2 312,9 | 63 021,6 | 1 664,9 | 61 671,2 | 1 777,2 | 71 543,8 | 1 834,6 |

Tabulka 1: průměrné naměřené hodnoty pro algoritmy uhel-otoceni, random-uhel-otoceni, random-360 a dlazdice

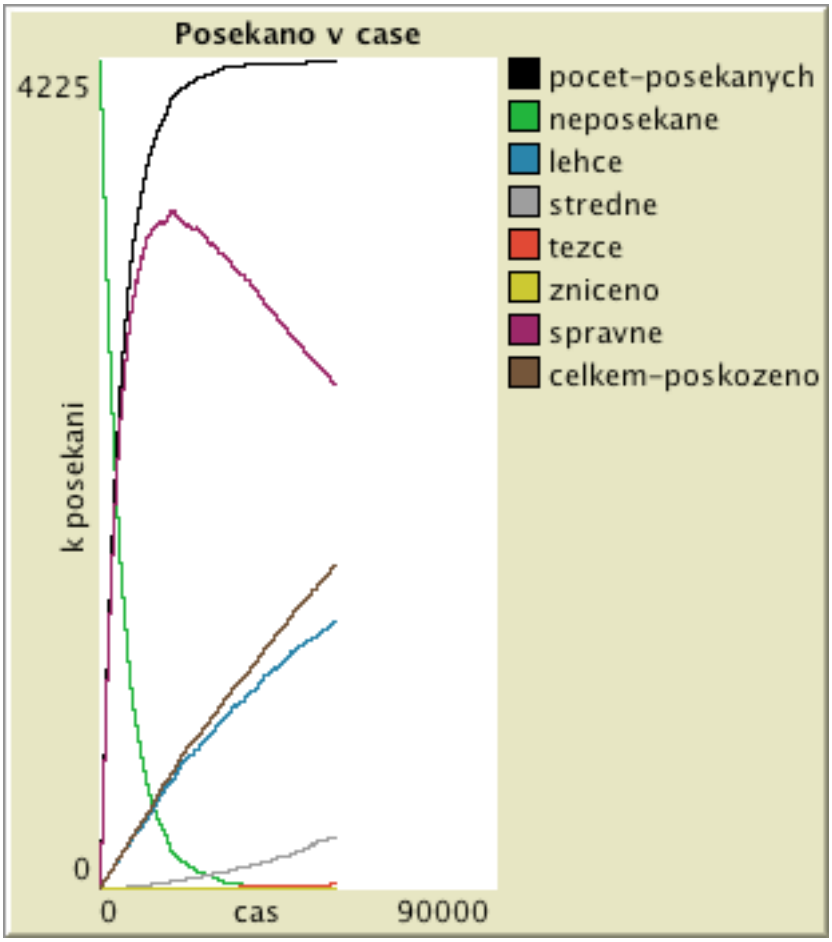
| B | uhel-otoceni | | random-uhel-otoceni | | random-360 | | dlazdice | |
|---|---|---|---|---|---|---|---|---|
| | ticks | škody | ticks | škody | ticks | škody | ticks | škody |
| 45 | 194 613,6 | 2 749,8 | 127 831,0 | 2 501,0 | 54 129,8 | 1 570,0 | 58 745,2 | 1 618,4 |
| 67 | 67 950,0 | 1 845,6 | 66 085,0 | 1 784,3 | 54 129,8 | 1 570,0 | 58 745,2 | 1 618,4 |
| 107 | 49 778,6 | 1 538,2 | 52 200,5 | 1 579,5 | 54 129,8 | 1 570,0 | 58 745,2 | 1 618,4 |
| 159 | 54 094,0 | 1 615,2 | 47 630,0 | 1 476,8 | 54 129,8 | 1 570,0 | 58 745,2 | 1 618,4 |
| avg | 91 609,1 | 1 937,2 | 73 436,6 | 1 835,4 | 54 129,8 | 1 570,0 | 58 745,2 | 1 618,4 |

Tabulka 2: průměrné naměřené hodnoty pro algoritmy uhel-otoceni, random-uhel-otoceni, random-360 a dlazdice, za přítomnosti lidí a zvířat

## 4.1 Algoritmus 1: uhel-otoceni

Tento velice jednoduchý algoritmus se ukázal býti naprosto nevhodným pro pohyb v prostoru bez lidí a bez zvířat. Protože jsme se pohybovali ve čtvercovém prostředí, úhly jako 45, 90, 30 atd. stupňů se vůbec nedokončily. Průměr 96 961 ticků je nejhorším výsledkem, době strávené na pozemku pak odpovídal i počet zničených dlaždic s trávou, v průměru 2 170 zničených dlaždic. A to nebyl do průměru započítán výsledek hraničící s nekonečnem v případě úhlu 45 stupňů.

Ve variantě B dosahoval algoritmus stejně špatných výsledků, 128 327 ticků a průměrné poškození 2 313.

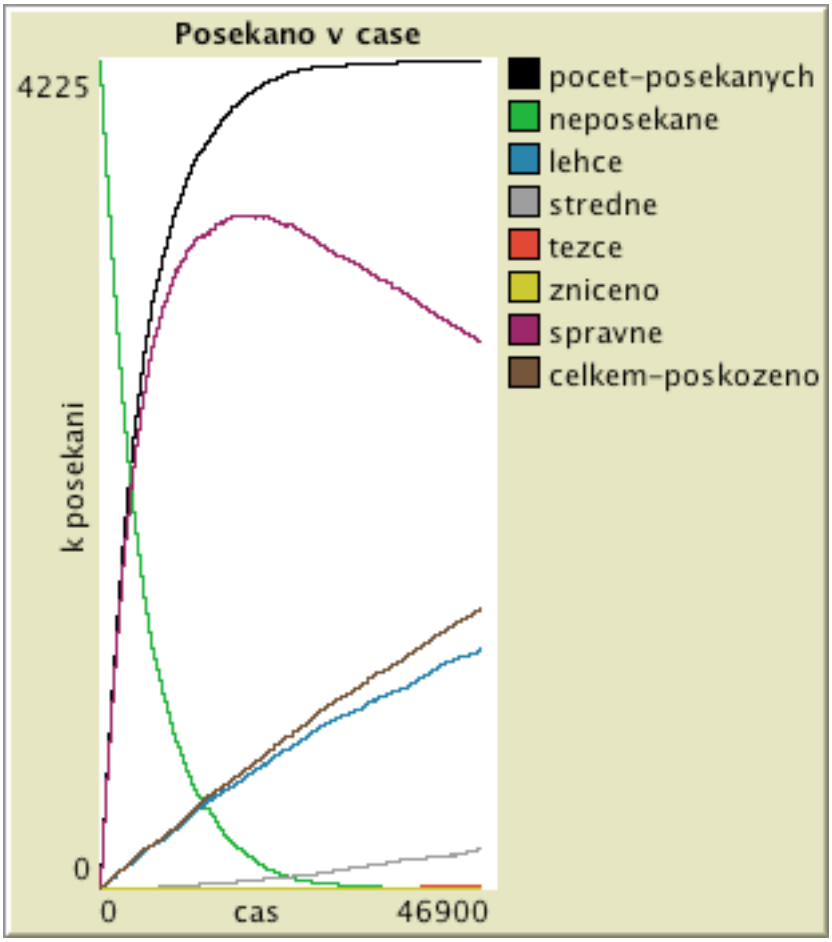



Obrázek 1: uhel-otoceni - vývoj proměnných v čase

Špatné výsledky jsou dány hlavně absencí náhody ve variantě A a spoléháním na srážku s některým z ostatních agentů ve variantě B.

## 4.2 Algoritmus 2: random-uhel-otoceni

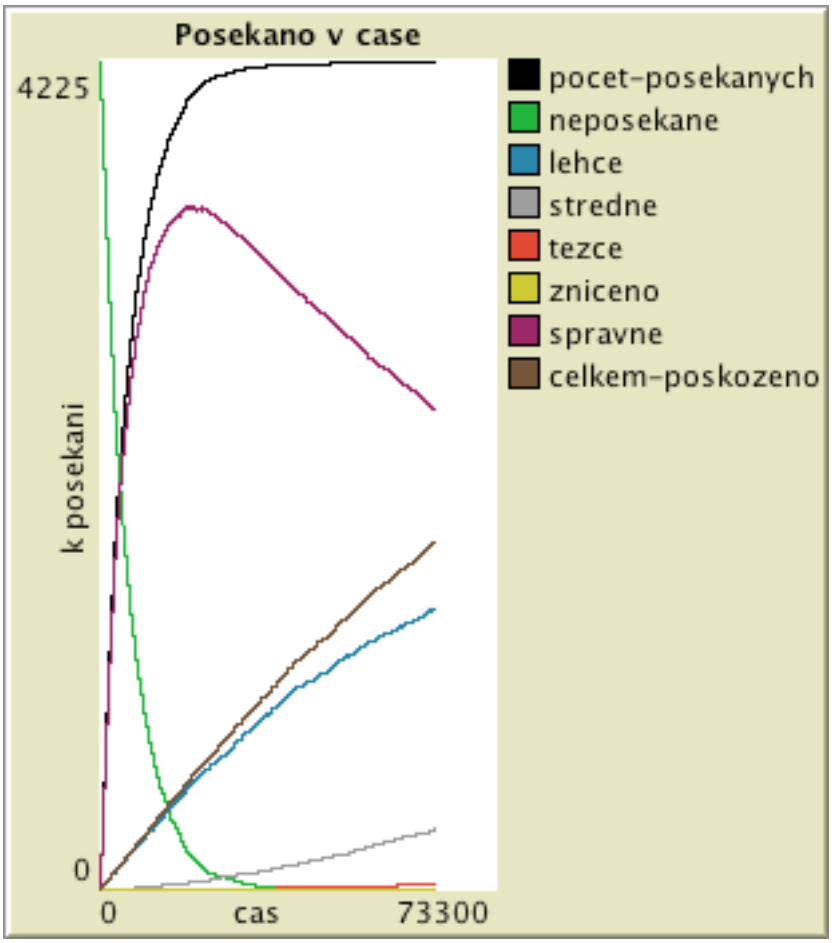
Bylo až s překvapením, že jsme mírnou změnou výpočtu úhlu otočení (použitím náhodného generátoru) dokázali dosáhnout až o 1/3 lepších výsledků než u předešlé varianty algoritmu.




Obrázek 2: random-uhel-otoceni - vývoj proměnných v čase

## 4.3 Algoritmus 3: random-360

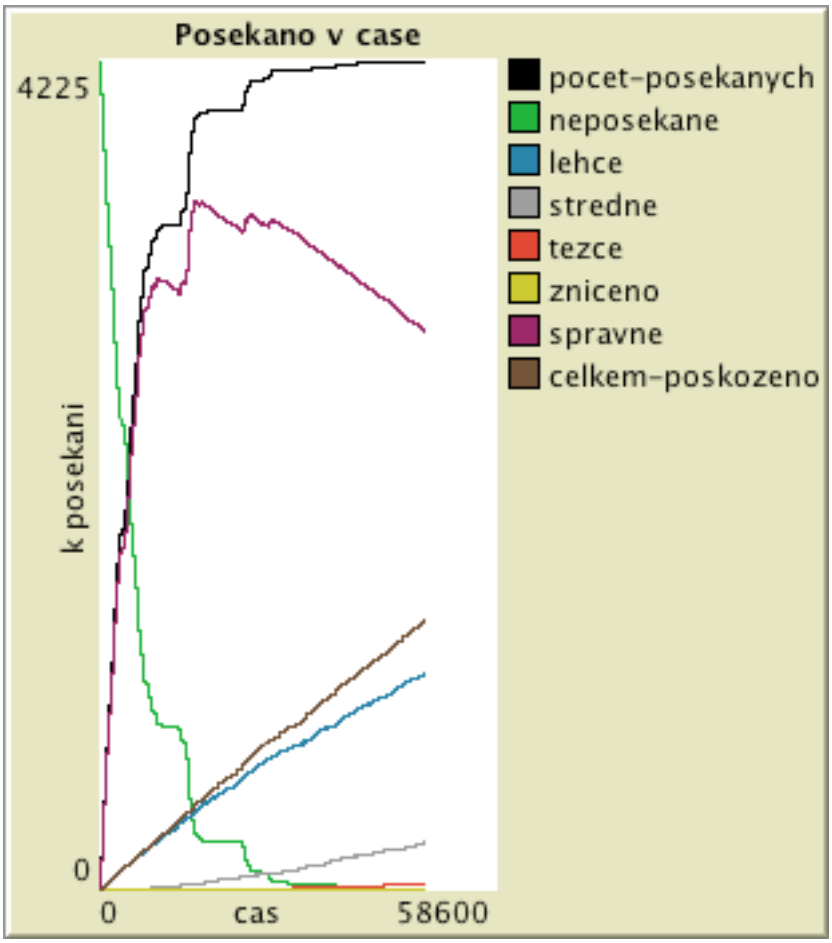
Poměrně jednoduchý algoritmus odrážející se zcela náhodně od překážek produkoval poměrně stabilní výsledky jak u varianty A, tak B. Z pokusů bylo zřejmé, že pohyb překážek na pozemku algoritmu svědčí, docházelo tak k časté změně směru a lepšímu pokrytí plochy. Zvolený úhel u tohoto typu pohybu nehrál roli.




Obrázek 3: random-360 - vývoj proměnných v čase

## 4.4 Algoritmus 6: dlazdice

Algoritmus dlaždice pro mě byl pravděpodobně největším zklamáním. Ačkoliv jsem do této metody vkládal hodně naděje, dosahovala pouze podprůměrných výsledků. Vinu přisuzuji části kódu metody, která reaguje na nenalezení dlaždice s neposekanou trávou, kde dochází k největšímu opotřebovávání okolních dlaždic.




Obrázek 4: dlazdice - vývoj proměnných v čase

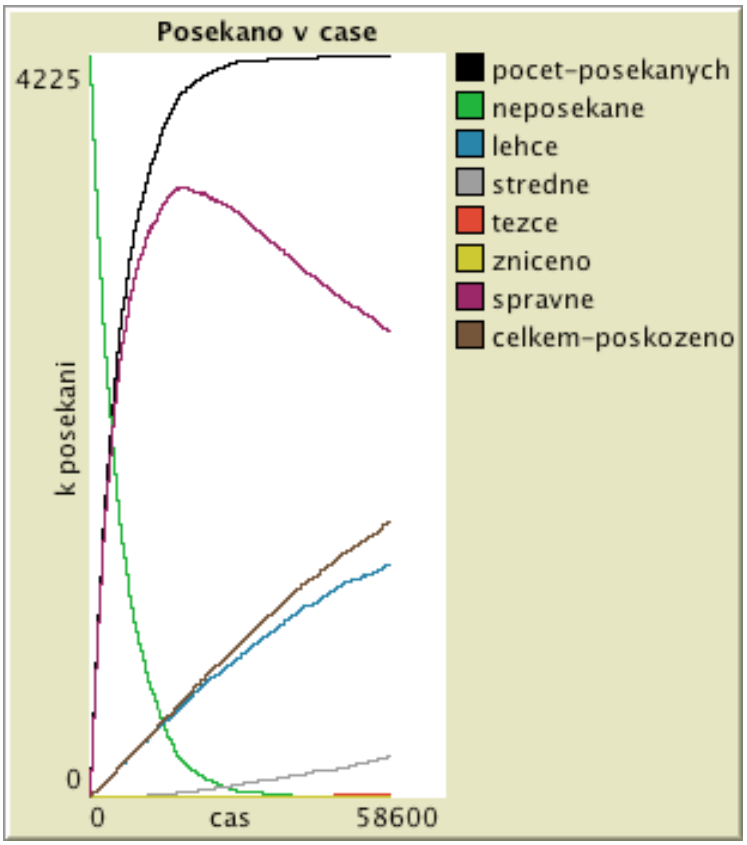
| A | kruh | MEAN 5 | kruh | MEAN 10 | kruh | MEAN 20 |
|---|---|---|---|---|---|---|
| | ticks | škody | ticks | škody | ticks | škody |
| 45 | 118 115,3 | 2 713,7 | 83 065,3 | 2 246,8 | 49 966,6 | 1 559,2 |
| 67 | 114 771,0 | 2 641,0 | 56 684,8 | 1 711,3 | 51 770,0 | 1 607,8 |
| 107 | 83 572,3 | 2 230,0 | 50 091,8 | 1 575,5 | 39 482,0 | 1 309,8 |
| 159 | 54 359,7 | 1 679,7 | 49 896,0 | 1 581,0 | 43 328,3 | 1 399,0 |
| avg | 92 704,6 | 2 316,1 | 59 934,4 | 1 778,6 | 46 136,7 | 1 468,9 |

| B | kruh | MEAN 5 | kruh | MEAN 10 | kruh | MEAN 20 |
|---|---|---|---|---|---|---|
| | ticks | škody | ticks | škody | ticks | škody |
| 45 | 96 218,0 | 2 469,0 | 58 980,0 | 1 797,0 | 44 204,7 | 1 427,0 |
| 67 | 94 712,5 | 2 402,5 | 49 217,3 | 1 575,3 | 43 919,7 | 1 396,0 |
| 107 | 103 717,8 | 2 507,8 | 50 249,0 | 1 578,3 | 47 098,3 | 1 497,0 |
| 159 | 60 997,5 | 1 797,0 | 46 062,8 | 1 449,5 | 46 490,3 | 1 463,7 |
| avg | 88 911,4 | 2 294,1 | 51 127,3 | 1 600,0 | 45 428,2 | 1 445,9 |

Tabulka 3: průměrné naměřené hodnoty pro algoritmus kruh

## 4.5 Algoritmus 4: kruh

Algoritmus kruh se projevil býti velice citlivým na nastavení jednotlivých parametrů. Překvapil mě hlavně fakt, že čím větší poloměr kruhu sekačka opisovala, tím lepších výsledků algoritmus dosahoval. Tyto rozdíly vyjadřuje v tabulce střední hodnota, kterou jsme postupně volili 5, 10 a 20.




Obrázek 5: kruh - vývoj proměnných v čase

Pravděpodobně nejpříjemnějším překvapením zde byl fakt, že na algoritmus téměř nemělo vliv nastavení okolních dynamických agentů. Jak ve variantě A, tak ve variantě B, dosahoval velice podobných výsledků.

| A | kruh-random-dojezd | MEAN 5 | kruh-random-dojezd | MEAN 10 | kruh-random-dojezd | MEAN 20 |
|---|---|---|---|---|---|---|
| | ticks | škody | ticks | škody | ticks | škody |
| **45** | 52 844,0 | 1 662,5 | 56 525,3 | 1 702,3 | 50 252,8 | 1 537,4 |
| **67** | 43 467,3 | 1 427,5 | 40 535,3 | 1 328,3 | 39 754,8 | 1 298,8 |
| **107** | 43 569,3 | 1 401,5 | 46 820,0 | 1 483,0 | 44 044,3 | 1 442,8 |
| **159** | 46 965,0 | 1 505,5 | 47 440,8 | 1 528,8 | 48 088,8 | 1 508,3 |
| | 46 711,4 | 1 499,3 | 47 830,3 | 1 510,6 | 45 535,1 | 1 446,8 |

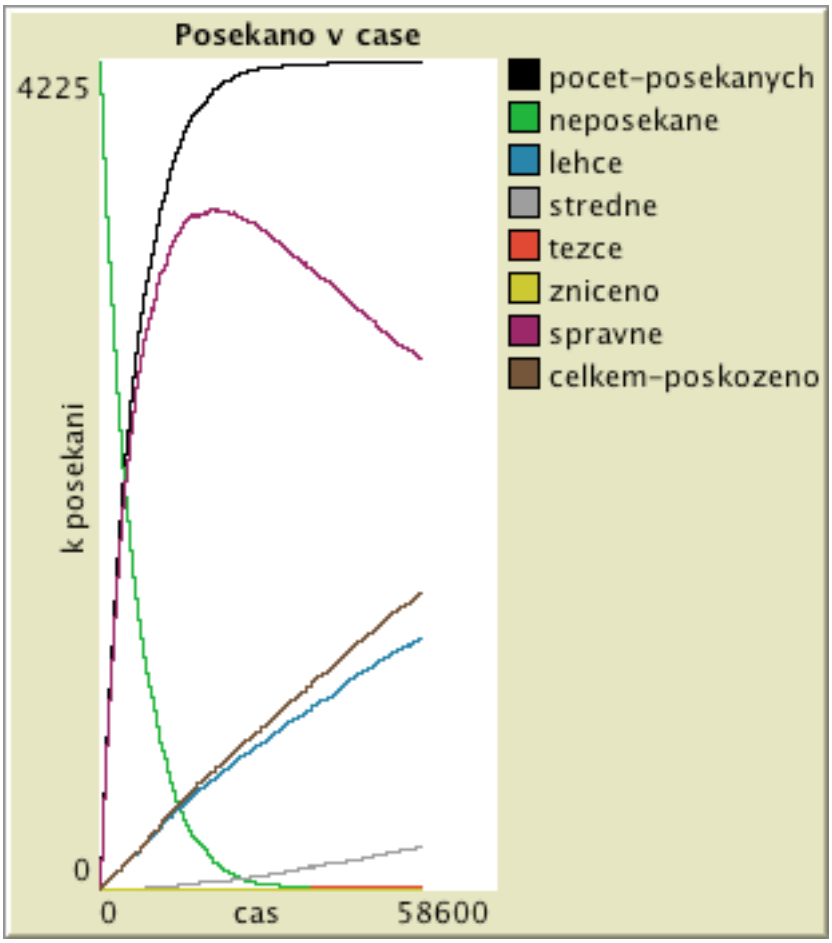
| B | kruh-random-dojezd | MEAN 5 | kruh-random-dojezd | MEAN 10 | kruh-random-dojezd | MEAN 20 |
|---|---|---|---|---|---|---|
| | ticks | škody | ticks | škody | ticks | škody |
| **45** | 53 708,2 | 1 673,8 | 53 830,5 | 1 690,0 | 47 466,5 | 1 481,0 |
| **67** | 49 225,8 | 1 523,6 | 49 863,0 | 1 542,0 | 55 382,3 | 1 689,3 |
| **107** | 48 536,0 | 1 530,5 | 47 734,6 | 1 488,4 | 44 771,2 | 1 456,0 |
| **159** | 44 255,5 | 1 388,3 | 46 337,5 | 1 477,3 | 46 090,0 | 1 468,4 |
| | 48 931,4 | 1 529,0 | 49 441,4 | 1 549,4 | 48 427,5 | 1 523,7 |

**Tabulka 4: průměrné naměřené hodnoty pro algoritmus kruh s random dojezdem**

## 4.6 Algoritmus 5: kruh s random dojezdem

Algoritmus pojmenovaný "kruh s random dojezdem" vznikl prakticky náhodou, čím více jsem ale o něm přemýšlel, tím lepší variantou se jevil. Vycházel jsem z teorie, že na začátku, kdy je velká většina pozemku neposekaná, se sekačce vyplatí jezdit v kruzích a systematicky tak posekat co nejvíce plochy. Po nějaké době by však bylo vhodné, aby sekačka změnila svůj styl pohybu a začala se po dlaždicích pohybovat více náhodně, aby tak dokázala pokrýt zbývající neposekané části.

Tuto teorii následně prokázala i samotná měření, algoritmus dosahoval druhých nejlepších a hlavně stabilních výsledků.



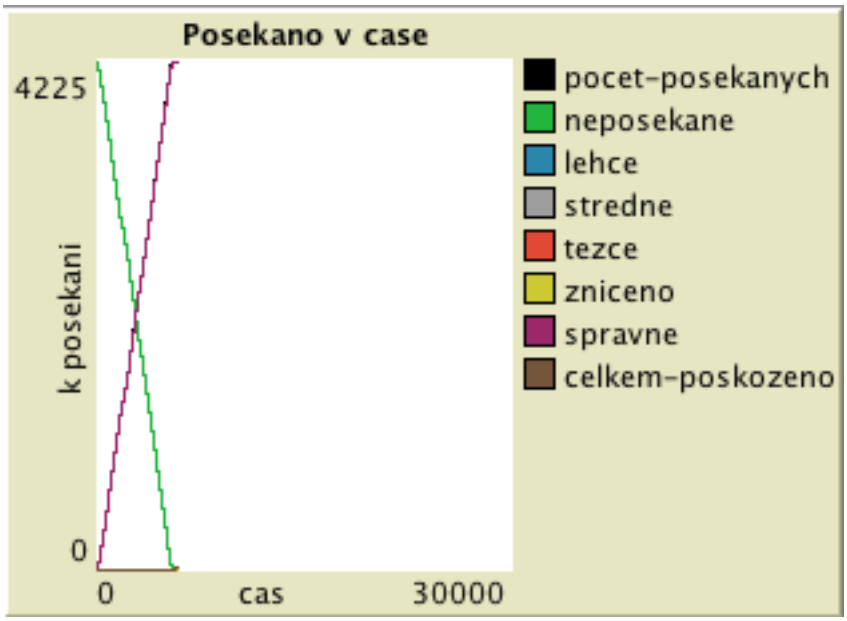
Obrázek 6: kruh-random-dojezd - vývoj proměnných v čase

## 4.7 Algoritmus 7: search

Nejpokročilejší algoritmus s názvem "search" jasně překonal všechny své předchůdce. Dosahoval 8-9x lepších výsledků než druhý nejrychlejší algoritmus "Kruh s random dojezdem" a škody na pozemku byly téměř minimální (v průměru okolo 23-27 poškozených dlaždic). Tyto skvělé výsledky jsou dány hlavně tím, že algoritmus dokázal analyzovat celé prostředí pozemku a efektivně vyhledávat neposekané kusy trávy. Narozdíl od ostatních algoritmů (s výjimkou algoritmu "dlaždice", kde byla viditelnost sekačky značně omezena), tak "search" netrávil čas neefektivním pojížděním po již posekaných částech spoléhajíc na náhodu.

| | search | |
|---|---|---|
| | ticks | škody |
| A (avg) | 6155,5 | 26,8 |
| B (avg) | 6017,4 | 22,5 |

Tabulka 5: Průměrné výsledky algoritmu search



Obrázek 7: search - vývoj proměnných v čase

# 5 Závěr

Cílem simulace bylo pozorovat vliv jednotlivých algoritmů a jejich nastavení na dobu trvání sečení a s tím související růst nákladů. Při volbě nejlepšího algoritmu bylo nutné zároveň sledovat poškození trávníku, jakého jednotlivé algoritmy dosahují.

Při plnění tohoto cíle jsme zjistili, že nejvhodnějším algoritmem bez analýzy širokého prostředí (spoléhající z velké míry na náhodu) je kruhový pohyb s přepnutím do náhodného módu, který podával stabilní výsledky při mnoha nastaveních a dosahoval nejlepšího času. Z výsledků jsme zároveň zjistili, že existuje přímá úměra mezi časem stráveným na pozemku a poškozeným trávníkem. Tento poměr se lišil pouze u algoritmu dlaždice, který měl na stejný čas méně poškozených dlaždic než ostatní. Byl by tedy vhodným řešením pro majitele, který neklade důraz na čas sečení, avšak disponuje ne příliš kvalitní sekačkou, která často trávník ničí.

V kategorii algoritmů, které dokáží pracovat s celým prostředím a aktivně umí vyhledávat neposekané dlaždice, exceloval algoritmus s názvem "search". Ten se prokázal jako vysoce efektivní, když dosahoval 8-9x lepších výsledků než ostatní metody pohybu, to vše při minimálním poškození trávníku.

Model nabízí řadu možností pro rozšíření. Kromě implementace nových prvků ovlivňujících model by například bylo vhodné sledovat, za jakou dobu sekačky dosáhnou pokrytí jistého poměru posekaných dlaždic, např. tedy jak rychle daný algoritmus zvládá posekat 90 % celé plochy. Další možností by bylo omezit dobu pobytu sekačky na pozemku a sledovat, kolik dlaždic za takto stanovenou dobu dokáže posekat.

# 6 Reference

[1] http://ccl.northwestern.edu/netlogo/.

# 7 Přílohy

zdrojové soubory modelu: *xdynl00-automaticka-sekacka.nlogo*