

## Průvodce nástrojem BehaviorSpace

Tento průvodce má tři části:

- Co je BehaviorSpace?: Obsahuje všeobecný popis nástroje, včetně principů, na jejichž základě funguje.
- Jak BehaviorSpace funguje?: Vysvětlí vám, jak se nástroj používá, a upozorní vás na jeho nejběžnější funkce.
- Pro pokročilé: Popisuje, jak používat BehaviorSpace z příkazového řádku nebo z vlastního kódu Javy.

### Co je BehaviorSpace?

BehaviorSpace je softwarový nástroj, jenž je součástí NetLoga a jenž umožňuje provádět s modely experimenty. Spustí daný model vícekrát, se systematicky odlišným nastavením a zaznamenává výsledky každého běhu. Tento proces se někdy nazývá „systematický průchod prostorem parametru“ (parameter sweeping) a slouží k tomu, abychom lépe prozkoumali možná chování modelu a určili, jaké kombinace nastavení způsobí zajímavé chování.

### Proč BehaviorSpace?

Následující postřehy nám napoví, proč tento typ experimentu potřebujeme. Modely mají často více nastavení, z nichž každé používá škálu hodnot. Společně vytvářejí tzv. prostor parametrů modelu, jehož rozměry jsou určeny počtem, nastavením a konkrétní kombinací hodnot v každém daném okamžiku. S každým jiným nastavením (a někdy i se stejným) modelu dostaneme naprosto odlišné chování v modelovaném systému. Jak byste tedy zjistili, jaké konkrétní uspořádání hodnot, resp. jaké typy uspořádání povedou k chování, jež vás bude zajímat? Čímž vzniká i otázka, na kterém místě v tom obrovském multidimenzionálním prostoru parametru bude model fungovat nejlépe?

Podívejme se na následující příklad. Potřebujete rychlou synchronizaci agentů v modelu Světlušky. Model má čtyři posuvníky – počet (number), délka cyklu (cycle-length), délka záblesku (flash-length) a počet záblesků (number-flashes) – které mají přibližně 2 000, 100, 10 a 3 hodnoty v tomto pořadí. Tzn. že existuje 2 000 x 100 x 10 x 3 možných kombinací hodnot na posuvnících! Nejrychlejší synchronizaci byste těžko zjistili tak, že byste kombinace zkoušeli jednu po druhé.

BehaviorSpace nabízí mnohem lepší způsob řešení. Určíte-li podmnožinu hodnot z rozsahu každého posuvníku, spustí nástroj model se všemi možnými kombinacemi těchto hodnot a u všech běhů zaznamená výsledky. Vytvoří vzorek prostoru parametru modelu, který sice není úplný, ale dostatečně zobrazí vznikající vztahy mezi jednotlivými posuvníky a chováním systému. Když jsou všechny běhy ukončeny, je vytvořena množina dat, již lze otevřít a prohlížet i v jiném programu, např. v tabulkovém procesoru, databázi či vizualizačním softwaru.

BehaviorSpace tak může být velice dobrým pomocníkem, když potřebujete prozkoumat celý prostor chování, jež model vykazuje.

### BehaviorSpace ve starších verzích NetLoga

Dřívější verze NetLoga (starší než 2.0) v sobě zahrnovaly starší verzi nástroje BehaviorSpace, která se však lišila v mnoha ohledech. Nebyla tak flexibilní, takže jste nemohli nastavit tolik typů experimentů.

Na rozdíl od současné verze však obsahovala funkci pro zobrazení a analýzu výsledků experimentu. U současné verze tato funkce chybí, předpokládáme totiž, že k analýze výsledků použijete jiný software. Do příští verze BehaviorSpace chceme znovu přidat možnost zobrazení dat a analýzy.

### Jak BehaviorSpace funguje?

Nástroj spustíte tak, že otevřete model a zvolíte z menu **Tools** položku BehaviorSpace.

### Správa nastavení experimentu

Otevřené dialogové okno vám umožní vytvářet, měnit, kopírovat, mazat a spouštět nastavení experimentu. Experimenty jsou uvedeny podle jména a podle toho, z jakých běhů modelu sestávají.

Nastavení experimentu je považováno za součást modelu NetLoga a ukládá se tedy společně s modelem.

Nové nastavení experimentu nastavíte stiskem tlačítka **New**.

### Jak vytvořit nastavení experimentu

V otevřeném dialogovém okně můžete určit následující informace – není ale nutné vždy zadat všechny údaje, některé části musejí zůstat prázdné nebo s výchozími hodnotami, to záleží na vás.

**Název experimentu (Experiment name).** V případě, že máte více experimentů, odlišíte je pomocí různých názvů.

**Obměna proměnných (Vary variables as follows).** Zde určíte, které nastavení chcete obměňovat, a jaké hodnoty mají být použity. Můžou to být proměnné posuvníků, přepínačů či roletek a jakékoliv globální proměnné obsažené v modelu.

Proměnné můžou zahrnovat i max-pxcor, min-pxcor, max-pycor a min-pycor, world-width, world-height a random-seed. Tyto údaje nejsou sice, přísně vzato, proměnné, ale v BehaviorSpace je můžete měnit, jako by byly. Změna rozměrů světa vám umožní prozkoumat, jaký má na model vliv velikost světa. Vzhledem k tomu, že hranice světa nejsou nutně definovány nastavením world-width a world-height, závisí jejich obměny na umístění výchozího bodu. Je-li výchozí bod ve středu, ponechá ho tam i BehaviorSpace, takže hodnoty world-width či world-height musejí být liché. Je-li jedna z hranic v nule, zůstane tam a posune se hranice opačná, např. když nastavíte svět s min-pxcor = 0 a max-pxcor = 10 a změníte ho pomocí world-width takto

```
[ "world-width" [11 1 14]]
```

zůstane min-pxcor v nule a max-pxcor je nastaven pro každé kolo na 11, 12 a 13. Jestliže není ani jedna podmínka pravdivá, znamená to, že výchozí bod není v centru ani na okraji světa, takže nemůžete měnit přímo world-height ani world-width, ale musíte obměňovat max-pxcor, max-pycor, min-pxcor a min-pycor.

Díky obměně random-seed můžete opakovat běhy se známým semínkem z generátoru náhodných čísel v NetLogu. Příkaz random-seed lze použít i v nastavení experimentu. Více informací o náhodných semínkách naleznete v podkapitole Náhodná čísla v Průvodci programováním.

Hodnoty lze nastavit buď tak, že vyjmenujete hodnoty, jež chcete užít, nebo specifikujete, že chcete použít každou hodnotu v daném rozsahu. Chcete-li tedy např. na posuvníku `number` použít každou hodnotu od 100 do 1 000 v krocích po 50, napíšete:

```
[ "number" [ 100 50 1000 ] ]
```

V případě, že chcete použít pouze hodnoty 100, 200, 400 a 800, zadejte:

```
[ "number" 100 200 400 800 ]
```

Všimněte si, že v druhém příkladě je méně závorek. Tím, že je uvedete, či nikoliv, říkáte BehaviorSpace, zda udáváte seznam jednotlivých hodnot nebo určujete rozsah.

Názvy proměnných musejí být v uvozovkách.

Můžete obměňovat, kolik nastavení chcete, pouze jedno nebo žádné. V nastaveních, která nezměníte, zůstanou současné hodnoty. Spuštění bez změny nastavení se hodí, když chcete spustit hodně běhů modelu s aktuálním nastavením.

Pořadí, v kterém se jednotlivé běhy uskuteční, záleží na pořadí, v jakém proměnné uvedete. Všechny hodnoty další proměnné budou vyzkoušeny dříve, než se modelování posune k dřívější proměnné. Pokud tedy chcete např. variovat  $x$  a  $y$  od 1 do 3 a  $x$  je uvedeno jako první, bude pořadí běhů následující:  $x = 1, y = 1$ ,  $x = 1, y = 2$ ,  $x = 1, y = 3$ ,  $x = 2, y = 1$  atd.

**Opakování (Repetitions).** Používá-li model čísla běhu, může se chování modelu v jednotlivých bězích lišit, i když je nastavení stále stejné. Chcete-li spustit model pro každou kombinaci nastavení více než jednou, zadejte zde číslo vyšší než jedna.

**Měření pomocí těchto reportérů (Measure runs using these reporters).** Zde určíte, jaká data chcete z každého běhu sbírat. Chcete-li např. zaznamenat, jak stoupá a klesá s jednotlivými koly populace želv, napište:

```
count turtles
```

Můžete zvolit jeden, více či žádný reportér. Pokud jich zadáváte více, musí být každý reportér uveden na zvláštním řádku, např.

```
count frogs
count mice
count birds
```

Běhy se uskuteční, i když nezvolíte žádný reportér. Tato možnost se hodí v případě, že chcete výsledky zaznamenávat jiným způsobem, např. pomocí příkazu `export-world`.

**Měření po každém kroku (Measure runs at every step).** NetLogo běžně měří běhy modelu v každém kroku pomocí reportérů zadaných výše. V případě, že jsou běhy modelu velmi dlouhé a stačí vám pouze data po dokončení každého běhu, odškrtněte tento rámeček.

**Příkazy k nastavení (Setup commands).** Těmito příkazy se bude spouštět každý běh modelu. Obvykle zadáte název procedury, která model nastaví, typicky PŘIPRAV (`setup`), je však možné zahrnout i jiné příkazy.

**Příkazy ke spuštění (Go commands).** Tyto příkazy budou probíhat stále dokola a posouvat tak model k dalšímu kroku. Obvykle zadáte název procedury START (`go`), ale můžete zahrnout i jakékoliv jiné příkazy.

**Podmínka k zastavení (Stop condition).** Způsobí, že běhy modelů budou různě dlouhé a vždy se zastaví, když bude splněna určitá podmínka. Předpokládejme např., že chcete, aby každý běh trval, dokud na světě nezůstanou žádné želvy. Napište:

```
not any? turtles
```

Chcete-li, aby všechny běhy měly stejnou pevnou délku, nechte políčko prázdné.

Běh se může rovněž zastavit, protože příkazy ke spuštění použijí příkaz `stop` (stejným způsobem se zastavuje trvalé tlačítko). Příkaz `stop` lze použít přímo v příkazech ke spuštění, nebo v proceduře, která je těmito příkazy přímo volána. (Účelem je, aby stejná procedura START fungovala jako tlačítko i v experimentu v BehaviorSpace.) Je důležité si uvědomit, že krok, v kterém je použito `stop`, se považuje za zrušený, takže z něj nejsou zaznamenány žádné výsledky. Test zastavení by se proto měl nacházet na začátku příkazů ke spuštění nebo procedury, nikoliv na konci.

**Závěrečné příkazy (Final commands).** Toto jsou příkazy navíc, které mají proběhnout až po dokončení běhu. Obvykle je toto políčko prázdné, ale můžete ho použít k volání příkazu `export-world` nebo jinému zaznamenání výsledku běhu.

**Časový limit (Time limit).** Umožní vám nastavit pevnou maximální délku každého běhu. Nechcete-li zadávat žádný limit, ale chcete, aby se délka běhů řídila podmínkou k zastavení, zadejte 0.

### Spuštění experimentu

Máte-li experiment nastaven, stiskněte tlačítko **OK** a potom tlačítko **Run**.

Budete vyzváni k volbě formátu, v jakém byste si přáli data z experimentu uložit. Data jsou zaznamenávána pro každý interval, běh nebo krok, a to v závislosti na nastavení volby Měření po každém kroku.

Ve formátu jednoduché tabulky (table format) jsou intervaly vypsány na zvláštním řádku a každý údaj ve zvláštním sloupci. Data jsou do tabulky zapisována vždy po dokončení běhu. Tento formát je vhodný pro automatické zpracování dat, např. pro importování do databáze nebo statistické aplikace.

Formát tabulkového procesoru (spreadsheet format) vypočte minimální, průměrné, maximální a finální hodnoty pro každý údaj a potom vytvoří seznam s každým intervalem na zvláštním řádku a údajem ve zvláštním sloupci. Tento formát je přehlednější než formát jednoduché tabulky, obzvláště importujete-li ho do tabulkového procesoru.

(U formátu tabulkového procesoru jsou data do souboru s výsledky zapsána až po dokončení celého experimentu. Do té doby jsou uložena v paměti, což může být problém u velkých experimentů, kde hrozí její nedostatek. Dále, pokud experiment cokoliv přeruší, např. běhová chyba, nedostatek paměti, zatuhnutí či dočasný výpadek proudu, nebudou výsledky nikde zapsány. U dlouhých experimentů vám tedy doporučujeme používat jak formát tabulkového procesoru, tak jednoduché tabulky, takže v případě, že se něco stane, budete mít alespoň tabulku s částečnými výsledky.)

Po zadání formátu výstupu vás BehaviorSpace vyzve, abyste zadali název souboru, do kterého mají být výsledky uloženy. Výchozí soubor má příponu .csv, můžete ho pojmenovat, jak chcete, ale nezapomeňte zachovat příponu, která značí, že se jedná o soubor typu „comma separated values“ (hodnoty oddělené čárkami, CSV). Tento formát obsahuje nezformátovaná data, jež lze načíst v jakémkoliv textovém editoru, ve většině tabulkových procesorech či databázových programech.

Objeví se dialogové okno označené **Running Experiment** (Běžící experiment). V tomto okně můžete sledovat, kolik běhů už bylo dokončeno a kolik uplynulo času. V případě, že jste pro měření běhů použili reportér a zaškrtnli jste políčko Měření po každém kroku, budou tyto hodnoty znázorněny ve formě grafu.

Běhy lze rovněž sledovat v hlavním okně NetLoga. (Pokud vám dialogové okno s průběhem experimentu překáží ve výhledu, jednoduše ho posuňte někam jinam.) Zatímco probíhá modelování, bude se aktualizovat zobrazení a grafy. V případě, že nepotřebujete aktualizace sledovat, odškrtněte tuto možnost v dialogovém okně s běžícím experimentem; experiment tím urychlíte.

Chcete-li experiment zastavit před dokončením, stiskněte tlačítko **Abort** (Zrušit). Nezapomeňte ale, že v tomto případě přijmete o všechna data, která se vytvářela pro formát tabulkového procesoru. Ve formátu jednoduché tabulky jsou výsledky zapisovány v průběhu experimentu, takže ztraceny nebudou.

Když skončí všechny běhy, je experiment dokončen.

### Pro pokročilé

#### Spuštění z příkazového řádku

Experimenty v BehaviorSpace lze spustit i jako „bezhlavé“ procesy, tj. z příkazového řádku bez grafického rozhraní uživatele (GUI). Tato možnost je vhodná zejména pro automatické spouštění na jednom stroji či jejich clustru.

K tomuto typu spuštění nepotřebujete ani nic programovat v Javě. Experiment můžete nastavit v uživatelském rozhraní a potom už ho jen spouštět z příkazového řádku, nebo můžete nastavení experimentu vytvořit či měnit přímo pomocí XML.

Nejjednodušší způsob je vytvořit nastavení experimentu předem v uživatelském rozhraní, takže je uloženo jako součást modelu. Nastavení experimentu uložené v modelu spustíte z příkazového řádku následujícím způsobem:

```
java -server -Xmx1024M -cp NetLogo.jar \
  org.nlogo.headless.HeadlessWorkspace \
  --model Fire.nlogo \
  --experiment experiment1"
```

(U souboru NetLogo.jar musí být podadresář lib obsahující potřebné knihovny. Soubor NetLogo.jar i lib najdete v NetLogu.)

Když experiment doběhne, jsou výsledky zaslány do standardního výstupu ve formátu tabulkového procesoru jako CSV. (Způsob, jak toto nastavení změníte, je popsán níže.)

Když spouštíte třídu Headless Workspace jako aplikaci, musí být vlastnost systému `java.awt.headless` pravdivá. Tím řeknete Javě, aby se spustila v „bezhlavém“ módu a umožnila běh NetLogo na strojích bez grafického zobrazení.

Aby Java mohla optimalizovat výkon aplikace na serveru, použijte parametr `-server`. Doporučujeme ho kvůli zlepšení výkonu používat ve většině situací.

Parametr `-Xmx` nastaví maximální limit dynamické paměti na jeden gigabyte. Jestliže tuto velikost nestanovíte, je určena výchozí velikostí virtuálního stroje, což často bývá příliš málo. (Jeden gigabyte je pro většinu modelů až příliš, můžete si nastavit jiný limit.)

Argument `--model` se používá ke zvolení názvu experimentu, jež chcete spustit. (V momentě, kdy v GUI vytváříte nastavení experimentu, přiřadíte mu název.)

Následuje příklad, který ukazuje další, volitelné argumenty:

```
java -server -Xmx1024M -cp NetLogo.jar \
  org.nlogo.headless.HeadlessWorkspace \
  --model Fire.nlogo \
  --experiment experiment2 \
  --max-pxcor 100 \
  --min-pxcor -100 \
  --max-pycor 100 \
  --min-pycor -100 \
  --no-results
```

Všimněte si použití volitelných argumentů `--max-pxcor`, `--max-pycor` atd., jež nastavují jinou velikost světa, než je uložena v modelu. (Hodnoty rozměrů světa je rovněž možné určit v nastavení experimentu, potom už je nepotřebujete uvádět v příkazovém řádku.)

Volitelný argument `--no-results` říká, že nemají být vygenerovány žádné výstupy. Hodí se v případě, že veškerý potřebný výstup je v nastavení experimentu proveden jiným způsobem, např. exportem souborů světa či zápisem do textového souboru.

Další příklad:

```
java -server -Xmx1024M -cp NetLogo.jar \
  org.nlogo.headless.HeadlessWorkspace \
  --model Fire.nlogo \
  --experiment experiment2 \
  --table table-output.csv \
  --spreadsheet spreadsheet-output.csv
```

Volitelný argument `--table <filename>` určuje, že výstup má být proveden ve formátu jednoduché tabulky a zapsán do daného souboru jako CSV. Jestliže je místo názvu souboru (filename) uvedeno `-`, pošle se výstup do standardního toku výstupu. Data se do tabulky zapisují po každém dokončeném běhu.

Volitelný argument `--spreadsheet <filename>` říká, že výstup má být uskutečněn ve formátu tabulkového procesoru a zapsán do daného souboru jako CSV. Jestliže je místo názvu souboru

(filename) uvedeno –, pošle se výstup do standardního toku výstupu. Data se nezapišou, dokud není dokončen celý experiment.

Je možné zvolit jak `--table`, tak `--spreadsheet`, v tom případě bude výstup zapsán do obou typů souborů

V případě, že není specifikován žádný formát, je výstup zaslán ve formě tabulky do standardního toku výstupu.

Následuje poslední příklad, který ukazuje, jak spustit nastavení experimentu, jež je místo v souboru modelu uloženo ve zvláštním XML souboru:

```
java -server -Xmx1024M -cp NetLogo.jar \
  org.nlogo.headless.HeadlessWorkspace \
  --model Fire.nlogo \
  --setup-file fire-setups.xml \
  --experiment experiment3
```

Jestliže soubor XML obsahuje více než jedno nastavení experimentu, musíte použít argument `--experiment`, abyste určili, jaký název nastavení se má použít.

V následující kapitole se dozvíte, jak vytvořit samostatný soubor XML s nastavením experimentu.

### Nastavení experimentů v XML

Ještě nemáme plně zpracovanou dokumentaci o tom, jak vytvořit nastavení experimentu v XML, ale pokud již jazyk XML znáte, mohou vám následující tipy dostatečně pomoci.

Struktura nastavení experimentu BehaviorSpace v XML je dána souborem „dokument type definition“ (definice typu dokumentu, DTD), který je uložen v NetLogo.jar jako `system/behaviorspace.dtd`. (Soubory JAR jsou komprimované, takže můžete DTD rozbalit pomocí nástroje Javy „jar“ nebo programem, který umí číst zazipované formáty.)

Jak nastavení v XML vypadá, zjistíte nejlépe tak, že si jich pár vytvoříte v grafickém rozhraní uživatele, uložíte model a potom se podíváte do výsledného souboru .nlogo v textovém editoru. Nastavení experimentu se nacházejí ke konci souboru .nlogo, v části mezi tagy `experiments`. Příklad:

```
<experiments>
  <experiment name="experiment" repetitions="10"
runMetricsEveryStep="true">
    <setup>setup</setup>
    <go>go</go>
    <exitCondition>not any? fires</exitCondition>
    <metric>burned-trees</metric>
    <enumeratedValueSet variable="density">
      <value value="40"/>
      <value value="0.1"/>
      <value value="70"/>
    </enumeratedValueSet>
  </experiment>
</experiments>
```

V tomto příkladě máme jen jedno nastavení experimentu, ale mezi začáteční a konečné tagy `experiments` jich můžete vložit, kolik chcete.

Když se podíváte do DTD souboru a na příklady nastavení vytvořené v GUI, bude vám už jasné, jak se používají tagy pro různé druhy experimentů. DTD určuje, které tagy jsou povinné a které volitelné, které lze opakovat a které ne apod.

Když je XML s nastavením experimentu uloženo v souboru modelu, nezačíná hlavičkou XML, protože je pouze součástí jiného souboru. Jestliže uložíte nastavení experimentu do zvláštního souboru, odděleně od souboru modelu, měl by mít příponu `.xml`, nikoliv `.nlogo`, a patřičné XML hlavičky:

```
<?xml version="1.0" encoding="us-ascii"?>
<!DOCTYPE experiments SYSTEM "behaviorspace.dtd">
```

Druhý řádek musí být zapsán přesně takto. V prvním řádku můžete určit kódování, třeba `us-ascii` nebo `UTF-8`. NetLogo však zpravidla nepodporuje jiné znaky než ASCII, takže změna kódování může být k ničemu.

### Řídící API

Jestliže vám BehaviorSpace nestačí, můžete využít naše řídící API, v němž můžete napsat kód v Javě, a tento kód bude řídit NetLogo. API vám umožní spouštět experimenty v BehaviorSpace z kódu Javy. Nebo můžete napsat vlastní kód umožňující řídit NetLogo přímo tak, aby vykonávalo stejné věci jako BehaviorSpace. Více podrobností o obou možnostech naleznete v kapitole [Průvodce řízením](#).

### Závěr

Nástroj BehaviorSpace stále ještě vyvíjíme. Budeme vám vděční, když nám napíšete na [feedback@ccl.northwestern.edu](mailto:feedback@ccl.northwestern.edu) a řeknete nám, jaké další funkce by se vám pro práci s ním hodily.



Copyright 1999-2009 by Uri Wilensky.  
Všechna práva vyhrazena.

Aplikace NetLogo, modely i dokumentace jsou šířeny veřejnosti zdarma pro účel tvorby a studia modelů. Software, modely a dokumentaci je možné pro studijní a výzkumné účely používat a měnit, a to za podmínky, že je výsledný produkt nabízen bezplatně a s uvedením informace o autorských právech a jménem původce na všech kopiích a související dokumentaci.

Pro jiné využití - než jsou výše zmíněné nekomerční způsoby - celku i jednotlivých částí (a to jak v původní, nebo změněné podobě) je třeba předem požádat o svolení od Uri Wilensky. Software, modely ani dokumentace nesmějí být užívány, přepisovány, ani upravovány jako součást komerčního softwaru nebo hardwaru bez předchozího získání licence od Uri Wilensky. Nezaručujeme kompatibilitu tohoto systému s jakýmkoliv jiným systémem a neposkytujeme žádné záruky.

Pro účely citování v akademických publikacích používejte tento odkaz:  
Wilensky, U. (1999). NetLogo. <http://ccl.northwestern.edu/netlogo>. Center for Connected Learning and Computer-Based Modeling. Northwestern University, Evanston, IL.