

# Softcomputing – projektová dokumentace

## Demonstrace průběhu řešení TSP pomocí mravenčích algoritmů

Vojtěch Dvořák (xdvora3o)

2. prosince 2024

### Úvod

Cílem tohoto projektu na téma *PSO, ACO či jiné přírodou inspirované algoritmy* bylo vytvoření aplikace pro demonstraci průběhu řešení problému obchodního cestujícího pomocí mravenčích algoritmů. Z možných variant zpracování se tedy jedná o první variantu (demonstrace algoritmu).

Kromě funkcionality demonstrační aplikace, jež je explicitně zmíněna v zadání (načtení souboru s daty, krokování, modifikace parametrů...), aplikace obsahuje také mnoho vylepšení, která v něm zmíněna nejsou (validace parametrů, možnost volby konkrétního algoritmu, nástroj pro analýzu výsledků...). Nejdůležitější funkcionalita aplikace je popsána v druhé části této dokumentace v rámci kapitoly 2. Zpracovány byly všechny varianty mravenčích algoritmů uvedené na přednáškách s výjimkou Rank-based Ant System. Aplikace může být kromě pouhé demonstrace také použita pro jejich porovnání a porovnání chování algoritmů s různými parametry.

## 1 Popis problému a implementace aplikace

### 1.1 Řešený problém

Vzhledem k tomu, že cílem bylo zejména demonstrovat fungování mravenčích algoritmů, aplikace řeší standardní problém pro tuto třídu algoritmů, a sice problém obchodního cestujícího (TSP). Jiné problémy lze pomocí těchto algoritmů řešit např. s využitím redukce. Tato problematika však není součástí tohoto projektu.

Mravenčí algoritmy TSP obecně řeší pomocí agentů (mravenců), kteří jsou umístěni na začátku iterace algoritmu do náhodných míst. Agenti poté přechází do dalších míst s pravděpodobnostmi, jež jsou ovlivněny vzdálenostmi mezi místy a intenzitou feromonových stop (příp. jinými faktory). Udržují si také seznamy již navštívených míst, aby se zamezilo jejich opakované návštěvě. Na konci iterace aktualizujeme nejlepší řešení na základě cest, kterými jednotliví agenti prošli, a aktualizujeme množství feromonu na cestách.

Popis fungování těchto algoritmů v předchozím odstavci je záměrně povrchní, neboť konkrétní mravenčí algoritmy se liší např. ve způsobu výběru následujícího místa nebo ve způsobu aktualizace intenzity feromonových stop. Algoritmy, jež jsou součástí aplikace, byly implementovány na základě přednášky v předmětu SFC.

### 1.2 Použité technologie

Projekt je vypracován v jazyce Python 3.12. Pro implementaci mravenčích algoritmů byla využita knihovna *numpy* a *scipy*. Aplikace s GUI využívá pro svoje fungování knihovny *tkinter* a *matplotlib*. Důraz byl kladen na využívání co možná nejmenšího počtu knihoven a modulů, které nejsou součástí samotného Pythonu.

### 1.3 Mravenčí algoritmy

Jak již bylo uvedeno výše, implementovány byly téměř všechny mravenčí algoritmy uvedené na přednáškách – Ant System, Ant Colony, Elitists Strategy, Ant Density, Ant Quantity a Min-Max Ant System. Cílem bylo umožnit uživatelům aplikace si algoritmus snadno vybrat, aby mohli sledovat odlišné vlastnosti a chování jednotlivých algoritmů. Protože se některé z těchto algoritmů liší od jiných uvedených pouze např. ve způsobu, jakým si mravenci vybírají následující místo, které navštíví, je zde s výhodou využita dědičnost. Třídy obsahující implementaci algoritmů tak mohou sdílet některé metody, které jsou napříč více algoritmy shodné, prostřednictvím společné nadtřídy (např. `AntSystemCommon`). Implementaci těchto tříd je možné nalézt ve zdrojovém souboru *ant\_algorithm.py*.

Základní jednotkou provádění implementovaných algoritmů jsou iterace (cykly). Jedna iterace zahrnuje náhodné umístění agentů, jejich průchod místy, určení nejlepší nalezené cesty, distribuce feromonu na cesty a vypařování feromonu. Běh algoritmu nemůže být pozastaven uprostřed iterace a tedy výsledná aplikace umožňuje krokovat s granularitou na úrovni těchto iterací nebo pozastavit provádění algoritmu až po skončení aktuální iterace. Provedení jedné iterace je realizováno voláním metody `make_step`.

OOP bylo využito také při implementaci agentů (mravenců) a při vytváření reprezentace prostoru, ve kterém se pohybují. Chování agenta je možné konfigurovat prostřednictvím parametrů. Kromě parametrů udávající váhu viditelnosti míst a váhu intenzity feromonové stopy při výpočtu pravděpodobnosti přechodu do nového místa, jsou v součásti parametrů také funkce, v rámci nichž se agent rozhoduje, kterou cestu zvolí a jak roz distributes nový feromon na cesty. Díky tomu je možné použít stejnou třídu reprezentující agenta pro všechny implementované algoritmy.

Prostor, v němž se agenti pohybují, je modelován pomocí tříd `Map` a `Place`. První z těchto tříd obsahuje matici vzdáleností mezi místy a matici aktuálního množství feromonu na cestách mezi nimi. Díky této reprezentaci feromonových stop je možné provádět úpravy intenzity feromonů pomocí operací sčítání matic a násobení matice koeficientem. Třída `Place` reprezentuje místo, kterým musí výsledná cesta procházet, a kromě souřadnic mohou objekty této třídy volitelně obsahovat pro větší přehlednost také název.

## 1.4 Uživatelské rozhraní

Uživatelské rozhraní zahrnuje interaktivní prvky a grafická zobrazení (např. nejkratší cesta, intenzita feromonových stop...). Vykreslování grafů je realizováno pomocí knihovny *matplotlib*. Vytvořené grafy jsou vkládány do GUI, do nějž jsou navíc přidány interaktivní prvky a textová pole s dodatečnými informacemi. Vykreslování grafů je poměrně náročné, a proto byla implementována možnost buď obnovovat grafy po každé aktualizaci nejlepšího řešení, nebo až při pozastavení a dokončení běhu algoritmu (výchozí volba).

Základem uživatelského rozhraní je hlavní okno obsahující všechny prvky pro základní ovládání aplikace. Pokročilejší funkce jsou pro větší přehlednost GUI umístěny do podoken, které je možné otevřít pomocí tlačítek v horní liště. Každé okno má dále svoji hierarchii prvků, jež je v některých případech za běhu aplikace modifikována, aby neobsahovala nerelevantní prvky (např. v historii běhů).

Aplikace využívá multithreading, aby nedocházelo k zamrznutí uživatelské rozhraní při provádění relativně výpočetně náročných mravenčích algoritmů. Vlákno zajišťující běh algoritmu je spravováno třídou `AlgorithmRunner`. Synchronizaci mezi tímto vláknem a zbytkem aplikace zajišťují objekty typu `Event` ze standardní knihovny *threading*.

## 1.5 Testování

Fáze testování probíhala s využitím datových sad třetích stran obsahující instance TSP<sup>1</sup>, u nichž bylo známo optimální řešení. Díky tomu bylo možné sledovat, zda algoritmy produkují rozumné výsledky, které se k ideálnímu řešení blíží. Výsledky závěrečného testování nad datovou sadou *westernsahara29\_27603* (29 míst, optimální délka cesty 27603) je možné najít v adresáři *history* a otevřít ji v historii běhů (viz kapitola 2.8).

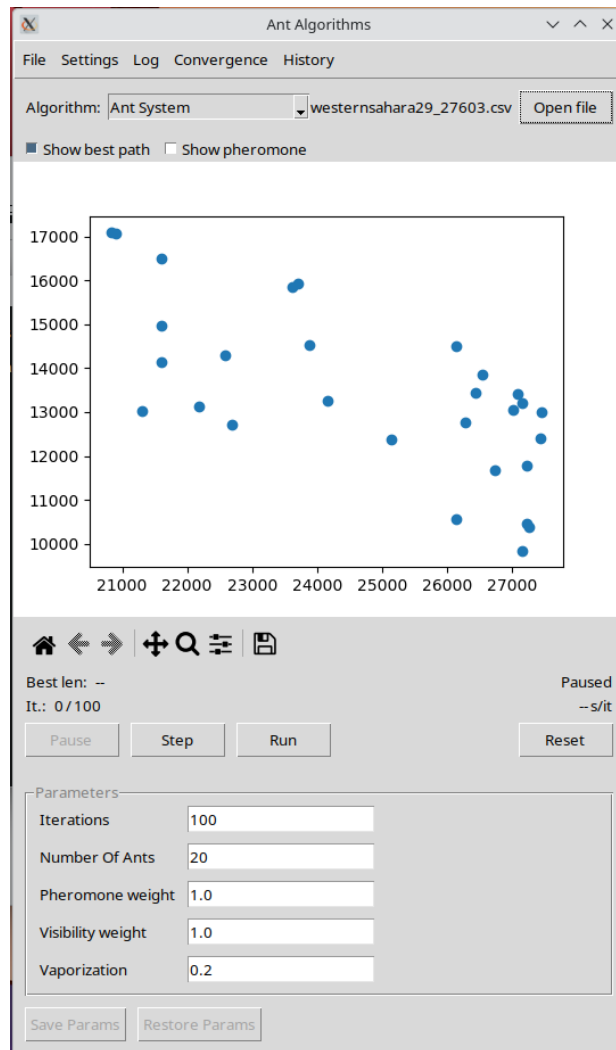
## 1.6 Omezení aplikace

Největší datová sada, která byla během testování použita, obsahovala 734 uzlů (míst)<sup>2</sup>. Rychlost algoritmu Ant System při počtu mravenců nastaveném na hodnotu 20 byla na běžném laptopu (AMD Ryzen 3500U, 16 GB RAM) v průměru 30 s/it. Optimalizace i v tomto počtu uzlů však probíhala bez dalších problémů.

Jelikož ale aplikace neumožňuje krokovat s menší granularitou než jsou jednotlivé iterace algoritmu, nezajišťuje aplikace v tomto případě přílišnou interaktivitu (např. doba odezvy při stisknutí tlačítka *Pause* je v nejhorším případě právě 30 s). Protože se však jedná o aplikaci, jež má sloužit primárně k demonstraci a experimentům, není ani očekáváno, že by byla využívána na takto velké instance TSP. Je vhodné pro účely demonstrace zvolit datové sady s méně než 100 místy, aby byla odezva malá a aby bylo možné algoritmus pohodlně krokovat. Poměrně komfortně bylo však možné krokovat provádění algoritmu i nad instancí TSP s 194 místy.

<sup>1</sup>Dostupné z: <https://www.math.uwaterloo.ca/tsp/world/countries.html>

<sup>2</sup>Datovou sadu je možné stáhnout zde: <https://www.math.uwaterloo.ca/tsp/world/uy734.tsp>



Obrázek 1: Hlavní okno aplikace s načtenou datovou sadou westernsahara29\_27603.

## 2 Uživatelský manuál

### 2.1 Závislosti

Program byl testován v prostředí interpretu jazyka Python ve verzi 3.12. Uživatelský manuál a instalační skript tedy předpokládá dostupnost interpretu pod příkazem `python3`. Dalšími závislostmi aplikace jsou knihovny: *matplotlib*, *numpy*, *scipy* a *tkinter*. Závislosti mohou být alespoň na referenčním stroji nainstalovány pomocí přiloženého skriptu *install.sh*.

Aplikace byla testována na referenčním stroji (Kubuntu) a také platformě Windows 10, pro tento systém však není v adresáři projektu dostupný instalační skript.

### 2.2 Instalace

Instalaci závislostí je možné provést prostřednictvím skriptu *install.sh* s administrátorskými právy:

```
sudo bash install.sh
```

### 2.3 Spuštění

Ke spuštění je opět možné využít jeden z přiložených skriptů, a sice *run.sh*:

```
bash run.sh
```

Skript v tomto případě spustí program s již načtenou datovou sadou z adresáře *datasets*. Alternativně je možné aplikaci spustit bez využití skriptu:

```
python3 app.py [<cesta-k-datove-sade>]
```

Aplikace má jeden volitelný argument, prostřednictvím kterého je možné v aplikaci otevřít soubor s datovou sadou. Tento argument však není nutné používat, neboť datovou sadu lze otevřít z GUI aplikace. Po spuštění aplikace se otevře její hlavní okno (Obrázek 1).

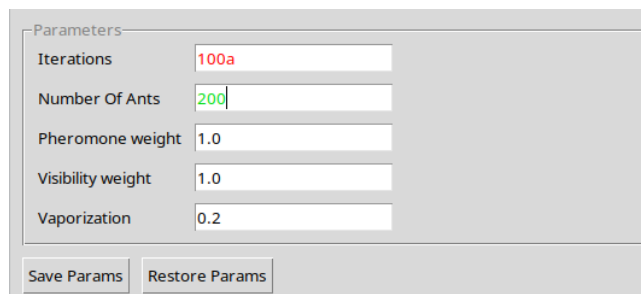
## 2.4 Načtení souboru s datovou sadou

V horní části hlavního okna vidíme tlačítko *Open file*. Po jeho stisknutí se otevře dialog, jenž uživateli umožní otevřít soubor s datovou sadou. Po vybrání souboru se zobrazí místa, která jsou v něm popsána, v grafu, přibližně uprostřed hlavního okna. Vedle tlačítka pro výběr datové sady můžeme vidět název souboru s právě načtenou datovou sadou. Pokud byl nad datovou sadou již spuštěn některý z algoritmů (viz dále), výběr datové sady má za následek restart aktuálního běhu.

Data je možné specifikovat ve formátu JSON a CSV. V souboru CSV se očekávají 2-3 sloupce bez hlaviček, z nichž první, volitelný, obsahuje název místa. Zbýlé dva sloupce obsahují souřadnice místa ve dvourozměrném prostoru. Příklady datových sad jsou dostupné v adresáři *datasets*. Ve formátu JSON jsou data specifikována podobným způsobem jako u CSV (viz *datasets*).

## 2.5 Výběr algoritmu a modifikace jeho parametrů

V horní části okna, vedle tlačítka pro výběr souboru s datovou sadou, najdeme rozklikávací seznam, z něž je možné vybrat konkrétní mravenčí algoritmus (výchozím algoritmem je *Ant System*). V dolní části hlavního okna najdeme sadu vstupních polí s hodnotami parametrů algoritmu. Konkrétní parametry se liší v závislosti na vybraném algoritmu. Například u algoritmu *Ant Colony* je mezi parametry také koeficient udávající pravděpodobnost upřednostnění exploitace před explorací. Hodnoty parametrů je pochopitelně možné měnit. Pokud byla hodnota parametru modifikována a její hodnota je validní, je označena zelenou barvou. V případě, že se jedná o nevalidní hodnotu, je označena červeně. Aby se změny parametrů projeví, je zapotřebí stisknout tlačítko *Save Params*. Alternativně je možné obnovit původní hodnoty parametrů stiskem *Restore Params*.



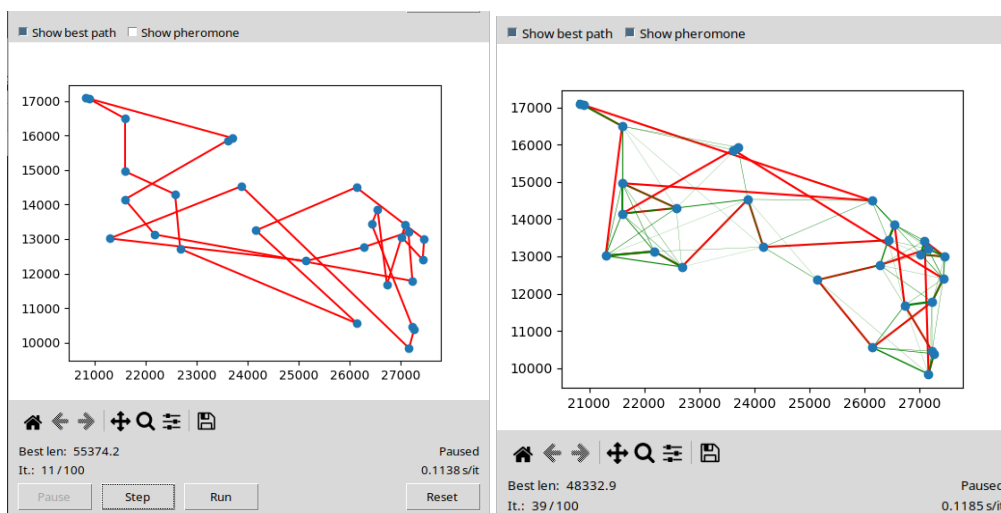
Obrázek 2: Rozhraní pro úpravu parametrů algoritmů. Zelená barva značí, že parametr byl modifikován. Červená barva značí, že parametr byl modifikován a že jeho hodnota není validní.

Uložení modifikované hodnoty parametru, obnovení původní hodnoty i změna algoritmu má za následek restart aktuálního běhu.

## 2.6 Ovládání běhu algoritmu a sledování průběhu

Algoritmus je možné spustit pomocí tlačítek *Run* a *Step*. První z těchto tlačítek spustí po zmáčknutí provádění algoritmu, které pokračuje až do jeho dokončení, nebo do chvíle, kdy je stisknuto tlačítko *Pause*. Tlačítko *Step* umožňuje krokování algoritmu po jednotlivých iteracích. Při jeho podržení přejde aplikace do stejného režimu jako u *Run* do chvíle, než je tlačítko *Step* uvolněno.

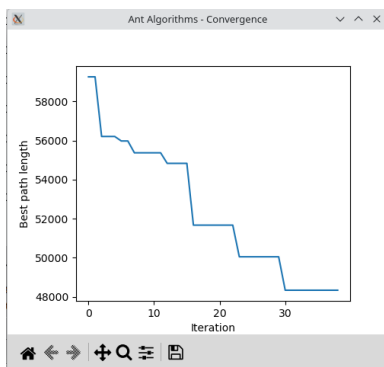
Sledovat průběh algoritmu je možné několika způsoby. V grafu, v hlavním okně vidíme dosud nejlepší nalezenou cestu (Obrázek 3). Graf se z důvodu zachování rychlosti provádění algoritmu aktualizuje, pokud je algoritmus pozastaven (při



Obrázek 3: Sledování běhu algoritmu v hlavním okně s vypnutým zobrazováním feromonu (vlevo) a se zapnutým zobrazováním feromonu.

krokování nebo pomocí tlačítka *Pause*), nebo pokud je dokončen<sup>3</sup>. Můžeme také pomocí checkboxu nad grafem zobrazit intenzitu feromonových stop. Jejich relativní intenzita je vyjádřena silou zelených čar v grafu. Pro větší přehlednost nejsou zobrazovány stopy, které patří mezi 10 % nejméně intenzivních. Přímou pod grafem je standardní ovládací panel knihovny *matplotlib*. Skrze něj je možné upravit zobrazení grafu (přiblížit některou oblast) nebo uložit obrázek s grafem.

Pod grafem můžeme vidět další informace o běhu algoritmu. Tato data jsou aktualizována v reálném čase. Najdeme zde délku nejlepší nalezené cesty, počet provedených iterací ku celkovému počtu iterací, stav běhu, a rychlost provádění algoritmu (v jednotkách sekunda na iteraci). Dále je možné sledovat průběh algoritmu pomocí zpráv v logu, jež najdeme v terminálu a v okně *Log*, které můžeme otevřít pomocí stejnojmenného tlačítka v horní liště hlavního okna. Do logu jsou zapisována také případná hlášení o chybách (např. chybný formát souboru s datovou sadou apod.). Log můžeme uložit do textového souboru pro pozdější analýzu (tlačítko *Save*).



Obrázek 4: Sledování průběhu optimalizace skrze graf s konvergenční křivkou.

Poslední možností, jak monitorovat průběh provádění algoritmu, je konvergenční křivka, tedy graf závislosti nejlepší nalezené cesty na iteraci (Obrázek 4). Toto okno můžeme otevřít pomocí tlačítka *Convergence* v horní liště hlavního okna. Pro aktualizaci tohoto grafu platí stejná pravidla jako pro graf v hlavním okně.

## 2.7 Ukládání a načítání parametrů

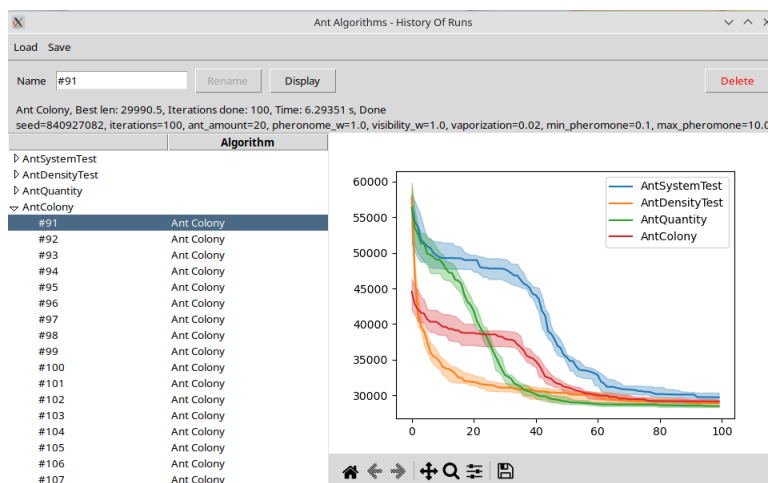
Aby bylo možné experimenty snadno replikovat, aplikace dovoluje uživateli ukládat a načítat parametry do aktuálního běhu skrze menu *File* v horní liště hlavního okna. Parametry mohou být uloženy včetně náhodného seedu, což způsobí, že po načtení uloženého souboru bude algoritmus probíhat identicky jako v době uložení. Pozor, po jakémkoliv restartu aplikace je ale vygenerován nový seed (pokud není seed zafixován v okně s nastavením). Pro účely otestování načítání

<sup>3</sup>Pokud je však zapotřebí graf aktualizovat při každém vylepšení nejkratší cesty, je tyto průběžné aktualizace možné zapnout v nastavení.

parametrů je možné najít uložené parametry v adresáři *parameters*.

## 2.8 Historie běhů a jejich analýza

Pokud má běh algoritmu jednu a více provedených iterací a dojde k restartu algoritmu, nebo je běh algoritmu dokončen, je běh uložen do historie běhů, kterou je možné zobrazit pomocí tlačítka *History*. V levé části okna s historií je seznam běhů od spuštění aplikace. Při výběru běhu je v horní části možné vidět jeho parametry, seed, celkový čas apod. Pomocí tlačítka *Display* můžeme zobrazit konvergenční křivku běhu. Vybereme-li běhů více, můžeme běhy seskupit a tím získat agregované výsledky a konvergenční křivku. Křivka v tomto případě označuje medián délky nejlepšího řešení v dané iteraci a zabarvená oblast označuje oblast mezi 1. a 3. kvantilem. Dále je možné historii uložit či načíst pomocí tlačítek *Load* a *Save*. Pro účely testování lze najít uloženou historii běhů v adresáři *history*.



Obrázek 5: Historie běhů – vlevo vidíme seznam provedených běhů a jejich seskupení, vpravo pak konvergenční křivky. V horní části vidíme detailní informace o právě vybraném běhu ze seznamu.

## 2.9 Pokročilá nastavení

Aplikace se snaží být do velké míry konfigurovatelná, a proto je možné v okně s pokročilými nastavením modifikovat seed, zafixovat ho (po restartech nebude generován nový) či změnit zobrazení v grafech. Můžeme zobrazit názvy míst v datové sadě (jsou-li poskytnuty), zobrazit délky úseků nejlepší nalezené cesty nebo hodnotu intenzity feromonu.

Prerekvizitou pro poslední dvě možnosti je však mít aktivováno zobrazení nejlepší nalezené cesty, resp. feromonu, pomocí checkboxu nad grafem v hlavním okně. Dále je možné zapnout také průběžné vykreslování grafů (konvergenční křivky i grafu v hlavním okně). To však může mít za následek zpomalení aplikace.

Dále můžeme nastavit opakování běhů algoritmu, což může být užitečné, pokud chceme provádět experimenty a statisticky je zpracovávat. Jelikož jsou mravenčí algoritmy stochastické je zapotřebí experimenty několikrát zopakovat, aby z nich bylo možné vyvodit závěry.

## Závěr

Hlavním výstupem tohoto projektu je aplikace pro demonstraci fungování mravenčích algoritmů nad TSP a pro provádění experimentů s nimi. Aplikace disponuje GUI, v rámci kterého je možné běh algoritmu monitorovat, modifikovat jeho parametry a provádět analýzu dokončených běhů v historii běhů.

Ačkoliv je pro demonstraci vhodnější aplikaci spouštět spíše nad menšími instancemi TSP. Experimentováno bylo ale i s poměrně velkým počtem míst v TSP (734), při kterém aplikace fungovala dle očekávání, sice správně a bez chyb, ale pomalu.

Aplikaci by bylo možné samozřejmě rozšířit v několika směrech. Bylo by možné přidat další varianty mravenčích algoritmů, umožnit další způsoby statistického zpracování experimentů apod. Díky oddělení GUI od samotných algoritmů by mohlo být zajímavé přidat do aplikace také jiné typy algoritmů (např. PSO). Reálné využití programu by mohlo být např. při výuce či jako nástroj pro hledání zajímavých parametrů mravenčích algoritmů.