

## Allgemein

Die Programmieraufgabe wurde in C# mit Verwendung von WPF gelöst. Es wurde auf Frameworks wie *MEF*, *Prism* oder *PostSharp* verzichtet. Dadurch ist die Anwendung sehr gut verständlich. Zur Lösung des Travelling Salesman Problems (TSP) wurde ein Simulated Annealing Algorithmus mit Standardwerten verwendet und zur Kontrolle Brute Force. Zudem bietet die Anwendung die Möglichkeit eigene Algorithmen zu implementieren, dazu muss von einer bestehenden Klasse (*Algorithm*) abgeleitet sowie eine Methode implementiert werden.

Bei Fragen stehen ich Ihnen gerne zur Verfügung: [kevin.wallis@hotmail.com](mailto:kevin.wallis@hotmail.com)

## Setup

Um das Programm ausführen zu können, muss das *.Net Framework* auf dem Computer installiert sein. Es gibt einen Ordner *Setup*, dort ist die Anwendung (**TravellingSalesman.exe**) und drei Testdateien (**TestA**, **TestB** und **TestC**) enthalten. Die Anwendung kann mittels Doppelklick geöffnet werden und sollte ohne Weiteres starten. Der Source-Code ist einem Extra-Ordner (*Source*) enthalten.

## Funktionalität

Wie bereits erwähnt wurde ein Simulated Annealing Algorithmus sowie Brute Force implementiert. Die Anwendung ist in Abbildung 1 ersichtlich. Insgesamt gibt es zwei Tab Pages, die Main Page, hier werden einerseits die Punkte verwaltet und andererseits die Optimierungen gestartet. In der zweiten Page sind die Statistiken ersichtlich und mittels Klick auf eine Statistik, wird der Pfad aufgezeigt (siehe Abbildung 2). Im Folgenden werden ein paar Punkte zur Hauptansicht gelistet:

- Es kann zwischen zwei Algorithmen gewechselt werden (Simulated Annealing und Brute Force).
- Neben dem Algorithmus werden Tourendistanz und Laufzeit angezeigt.
- In der Liste können die Punkte bearbeitet werden, solange die Anwendung nicht läuft. Jede Bearbeitung wird sofort in der Grafik aktualisiert.
- Der ausgewählte Punkt aus der Liste wird rot in der Grafik dargestellt.
- Mittels Klick auf den Mülleimer (rechts oben), wird der ausgewählte Punkt gelöscht und die Grafik neu verbunden.

- Durch die Eingabe einer X und einer Y Koordinate können neue Punkte definiert werden (der Button mit dem Plus bestätigt das Hinzufügen). Hierbei ist zu beachten, dass nur Zahlen zwischen 0 und 500 (inklusive) eingegeben werden können, Buchstaben werden nicht angenommen.
- Ein Ausführung kann mit Klicken auf den grünen Knopf gestartet werden.
- Während der Ausführung wechselt der grüne Knopf zu rot und bei erneutem Klicken bricht die Optimierung ab.
- Jeder Abbruch einer Ausführung erzeugt eine Statistik, gleiches gilt auch, wenn der Algorithmus selbstständig beendet wird.
- Die Anwendung bietet die Option eine bestehende Landschaft (Punkte) zu speichern und später wieder zu laden.

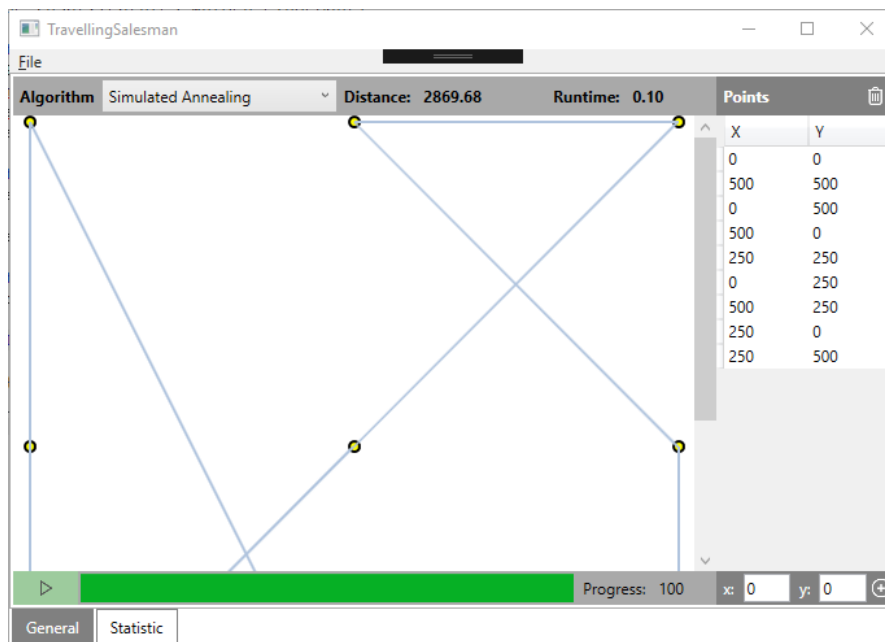
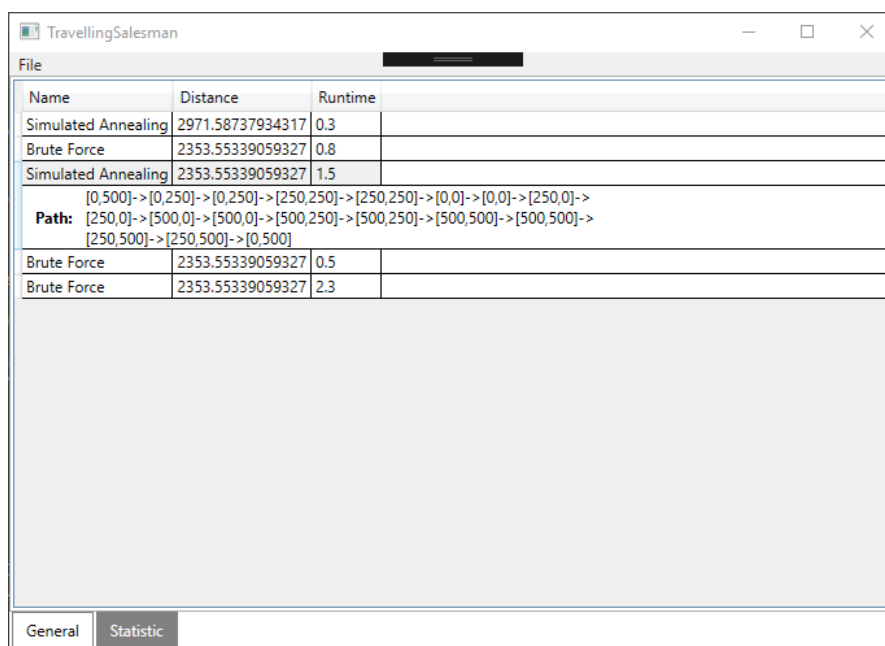


Figure 1: Main Page



| Name  | Distance         | Runtime |
|---|------------------|---------|
| Simulated Annealing   | 2971.58737934317 | 0.3     |
| Brute Force   | 2353.55339059327 | 0.8     |
| Simulated Annealing   | 2353.55339059327 | 1.5     |
| <b>Path:</b> [0,500]->[0,250]->[0,250]->[250,250]->[250,250]->[0,0]->[0,0]->[250,0]->[250,0]->[500,0]->[500,0]->[500,250]->[500,250]->[500,500]->[500,500]->[250,500]->[250,500]->[0,500] |                  |         |
| Brute Force   | 2353.55339059327 | 0.5     |
| Brute Force   | 2353.55339059327 | 2.3     |

General Statistic

Figure 2: Statistics Page