Ye. Gayev, Dr, V. Kalmikov
(National Aviation University, Ukraine)

The Travelling Salesman Problem in the engineering education programming curriculum

The paper explains shortly the famous Traveling Salesman Problem, develops GUI MATLAB-program for its solution by "Brute-force algorithm" and suggests to include this in standard curriculum of Programming for students in aeronavigation area.

Mastering complex algorithms and their programming realization became an important part of modern education on each engineering specialties. There are two motivations for this: students get a powerful tool that facilitates learning almost all other disciplines, and master the instrument for developing computer programs for their future professional practice. Educators are obliged to look for new benchmarks and appropriate computerized platforms to allow their learning computer science as effective as possible. We find it not effective the common approach to learn particular popular languages that start from the famous standard program *Hallo World*!. Instead, we suggest learning mathematical platforms like Wolfram Mathematica, Maple, MATLAB etc. that focus on algorithmization processes providing ready tools for basic mathematical problems rather than peculiarities of programming language. We accept the last one, the MATLAB, as one of the most appropriate for students in aeronavigation. As such, we follow to revolutionary educational approach "*Learn things by imaginable computer experiments*" suggested in recent book [1].

Such an educational approach requires just another means to attract students, and also a totally different collection of exercises and practicing tasks. It is to background in this paper that such a difficult problem as Traveling Salesman Problem (TSP) may serve as one of them. The latter combines all the cognitive content required for mastering algorithmization and programming with the practical direction to aeronavigation area. We discuss in this paper the most direct solution of the TSP in MATLAB along with fundamental subprograms required for it.

Mathematical formulation of the Traveling Salesman Problem (TSP) was initially given in 1800s by W.R.Hamilton. It sounds in the following way: a "salesman" from the city 0 should visit all the other n cities named as 1, 2, . . . , n only once and come back to 0; among $N=n!$ possible routes the shortest is to be found. Despite the simplicity of the formulation and the significant "age" of the problem, it turned to be one of most complicated in the discrete mathematics and optimization theory up to nowadays.

Direct solution of the TSP lies in the "honest" search of all the possible routes, accurate calculation their length and comparing them until all the variants are checked. Existence of a solution is, theoretically, thus evident. With the grows of n, the number of cities, it turns however that the time, required for enumeration of all possibilities, grows drastically so that the solution cannot be achieved during the reasonable time period. As an example: the time $T(100)$ is, say, equal to 1 minute for n=100; the time for n=101 will be $T(101)=101*T(100)$, i.e. about 100 minutes; moreover, the next case for n=102 cities will consume the time $T(102)=102*T(101)$, i.e. about 100^2 minutes, or 167 hours, or about 7 days. That is why this "naive" way was called “brute-force algorithm”, and efforts were paid into invention of more heuristic but realistic algorithms. Several such algorithms were elaborated, that found their recent applications in effective popular automotive GPS navigators [2]. However, the TSP still remains unsolved “in full”. The problem’s state of the art has been completely described in popular resources as [3,4]. It is not surprising that the TSP is topical for aeronavigation as well.

The Traveling Salesman Problem has already been realized in MATLAB. One should simply run in the Command Window

>>travel (1)

(the symbol >> denotes relation to the Command Window of MATLAB). Graphical User Interface (GUI) that appears is shown in the Figure 1A along with solution for n=50 cities on American map. It is imperfection of this program that enumeration of routes currently in consideration is too fast to follow them and to understand the process. Why do we suggest such TSP problem for engineering programming education just when it has been solved and realized in such and in similar demonstrations? Our answer is: 1. To present to students such a practical task on their major in aeronavigation area; 2. To encourage them to their original research and development, and 3. To allow research of their own by means of programming, especially testing the program efficiency (section IV,F). It is to account as well that the MATLAB-program has other weak points and is oriented to USA rather than to our country Ukraine.

Graphical appearance of our program called *SalesMan.m* has been shown in the Figure 1,B. It uses the map of Ukraine. Issuing *SalesMan* in the MATLAB Command Window like (1) leads to the Graphical User Interface (GUI) shown. The button “*Help*” provides a short explanation window *2a* on the right top. One could choose the number of cities in the narrow List Box window *3* to consider, n. For demonstration purpose, it is prescribed for only n of 5, 10, 15 and 20. When chosen, the button “*Generate cities*” is to be pressed, and corresponded number of Ukrainian cities becomes labeled as green points, see Fig. 1B for n=20. Pressing then the button “*Optimize distances*” *4b* starts process of possible routes enumeration, and each route under consideration may be observed by user. At the end, the final route with the least length is demonstrated. As told, we “honestly” enumerate all the possible routes what is called the “Brute-force Method”. We find it methodologically correct to research this method first; students will have a comparison base if go on with other methods, more heuristic.

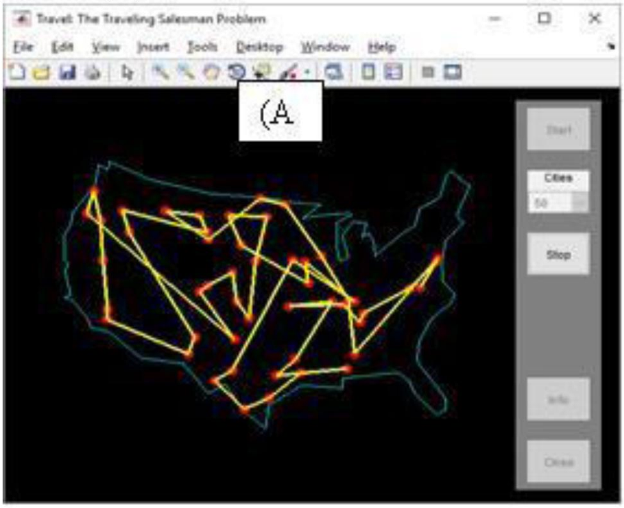
When the final programming product has been highlighted in such a way, we can start explanation how to go to such result. To achieve it, one needs to split the whole problem to several tasks, subtasks and subprogram. We describe most important of them in the next section. There are few relative simple subprograms among them, but there are also some especially important ones that form an educational and intellectual background of the profession of programming.

First, one needs to create Graphical User Interface (GUI) of the program where all its functionality should be foreseen by means of windows for inputting and outputting information and by buttons that start execution of certain tasks. One of latter is a procedure how to choose particular cities. Secondly, a subprogram should be elaborated that enumerates all possible routes between the cities chosen (the brute-force method). Thirdly, all the routes generated are to be "passed by the program" and the length of them calculated. It would be visual to demonstrate on the map any route currently in consideration and control duration of the visualization. Finally, the route with the least length is to be returned as the final problem solution for the city collection chosen.

There is a special program in MATLAB, *guide*, that helps in creation Graphical User Interface. Choosing standard GUI-elements such as Static Text *1*, Edit Text, List Box *3* or Pop-Up Menus, Push Buttons *2* and *4* and Axes *5* etc. from special visual environment, programmers are able to create variety of pleasant GUIs. For our GUI in Fig. 1,B these elements have been displayed under the numbers specified. The program is saved in MATLAB workspace in two files *SalesMan.fig* and *SalesMan.m*. The first one is a binary file keeping all the visual information (GUI element positions, their colors, fonts etc.), but the second is the text with the future program. It contains function signatures for all the GUI-elements yet empty and to be programmed in next sections.

What is to stress here is projection of picture with a map of Ukraine to the GIU axes *5*, Fig. 1B. This is an undocumented MATLAB' feature that will be explained in a separate publication.

The simplest GUI-elements of the GUI-program Fig. 1B are the Static Texts *1* and the Push Button *2*. The first ones are unchangeable and serve mainly for inscription and titling. The Push Button *2*, through corresponding function within the *SalesMan.m*, refers to only ready MATLAB-command *helpdlg*( ) that displays the Help-information labeled as *2a*. To the List Box *3* corresponds another function in the program-file *SalesMan.m* that supplies the integer *n* obtained to one more function of the Push button *4a* "*Generate cities*". The latter starts a simple logic that chose one of totally four prepared collections of cities along with their coordinate pairs $\{x_i, y_i\}$ on the Ukrainian map. The final route will be colored green and its length, the least among others, will be displayed in the Edit Text window *6*.


Travel: The Traveling Salesman Problem
File Edit View Insert Tools Desktop Window Help
(A
Start
Cities
50
Stop
Info
Close

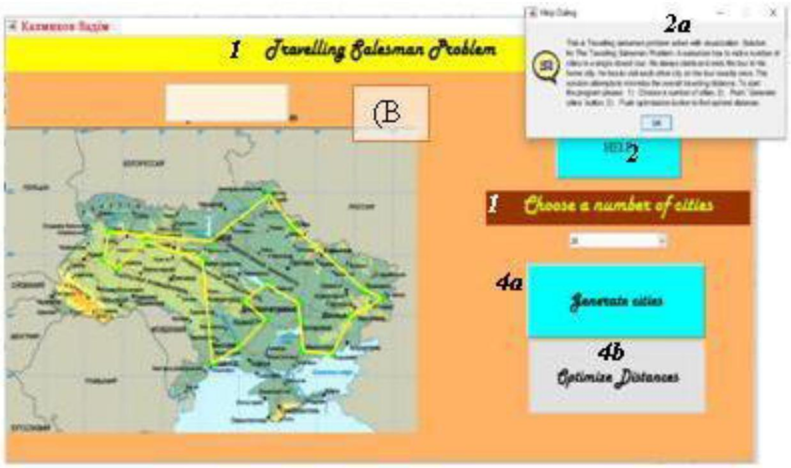


Fig. 1: (A) TSP GUI in MATLAB; (B) GUI of our MATLAB-program with the map of Ukraine

Behind this Button in the file *SalesMan.m* lies the most profound algorithm portion that starts all possible enumerations of city collection chosen and, simultaneously, calculation of their route length. At the same time the route under consideration is ruled thick yellow to visually follow and observe the algorithm action. This consideration of all the roots is called the "Brute Force Algorithm". It may be partially shortened in time if it breaks out any current route with the length that exceeds the length of previous routes already examined. Different approaches are mentioned in section *F*.

The most important part of the algorithm described is generation of permutations of cities from the collection chosen.

Many of algorithms and subprograms required here may devote special attention. Generation of permutations is particularly difficult to students and is described in more details below. Solution suggested below is based on recursion what is one of key stone algorithms in Computer Science.

Say, we need to get all permutations of numbers 1, 2, . . ., *n*, what means that function *T*=*Permutations*(*n*) depends on *n*. In the simplest case *n*=1 the list of permutations *T* contains only 1. If *n*=2 the *Permutation*-program returns *T* as a matrix [1 2]. Similar, *Permutation*(3)=[1 2 3; 1 3 2; 3 1 2] , i.e. new element 3 situates within all previous matrix rows.

As the result, students have a pleasant up-to-date graphical program *SalesMan* that realizes the real logistics problem in aeronavigation area. It is worth to provide its wide investigation, and particularly the time it consumes. They estimate that the *Time* is proportional to *n*!, i.e. it grows drastically huge, see section *II*. It is to motivate them to look for other algorithms able to reduce the *Time* significantly. A number of heuristic approaches were suggested in five last decades [2,3] but the TSP remains far from complete solution.

Conclusion

Despite the Traveling Salesman Problem (TSP) is one of the most difficult in discrete mathematics, its synopsis was shortly suggested to include in educational courses for students in aeronavigation area. MATLAB was considered as one of most appropriate mathematical workbenches for this. The program *SalesMan* was developed and briefly explained as a single GUI-program that manages several other programs and algorithms to realize "brute-force method" in the search of shortest route between given number of cities. Ukrainian map and several collections of Ukrainian cities were used to demonstrate the program. Several programs employed belong to key stone algorithms of the Computer Science.

References

1 Wolfram S. A new kind of science.--1213 pp.
2 https://en.wikipedia.org/wiki/GPS_navigation_device An overview of GPS technique.
3 https://en.wikipedia.org/wiki/Travelling_salesman_problem A full and modern overview of the Travelling Salesman Problem (TSP).
4 https://ru.wikipedia.org/wiki/Задача_коммивояжёра A rather detailed description of the TSP art state.