Russian literature, the traveling salesman problem, sanctions, SMath, Maple...

This chapter describes a lesson in STEM technology at the intersection of mathematics, computer science, literature, Moscow studies and even political science. A technique for solving the traveling salesman problem by the nearest neighbor method in the environment of the mathematical program, SMath, is described.

Keywords: graph theory, traveling salesman problem, nearest neighbor method, SMath, Maple, programming

Chekhov has a rather funny story: "New Year's Torture". The text is short—it is worth reading [1] or listening to [2], and then returning to this article. The story describes how a scandalous wife kicks her husband out of the house to visit relatives and friends—to congratulate them on the New Year: "*You thought not to make visits!*". To do this, it was necessary to drive around Moscow along the following route: Zubovsky Boulevard (the start is the place where the hero of the story lived), Red Barracks in Lefortovo, Khamovniki, Nizhny Novgorod Station[1], Krestovskaya Zastava, Kaluga Gates, Sokolnitskaya Grove, returning home to Zubovsky Boulevard.

The author tried to order such a circular trip using one of the internet applications for ordering a taxi. Here's what he did—see Figure 1.

[1] It was a temporary station for the Moscow-Nizhny Novgorod railway. It was later extended to the Kursk railway station, and the Nizhny Novgorod railway station itself was demolished. From it remained the name of Nizhegorodskaya street. Krestovskaya Zastava was in the area of the current Rizhskaya metro station, and the Kaluga Gate was in the area of the Oktyabrskaya metro station, which used to be called Kaluzhskaya.

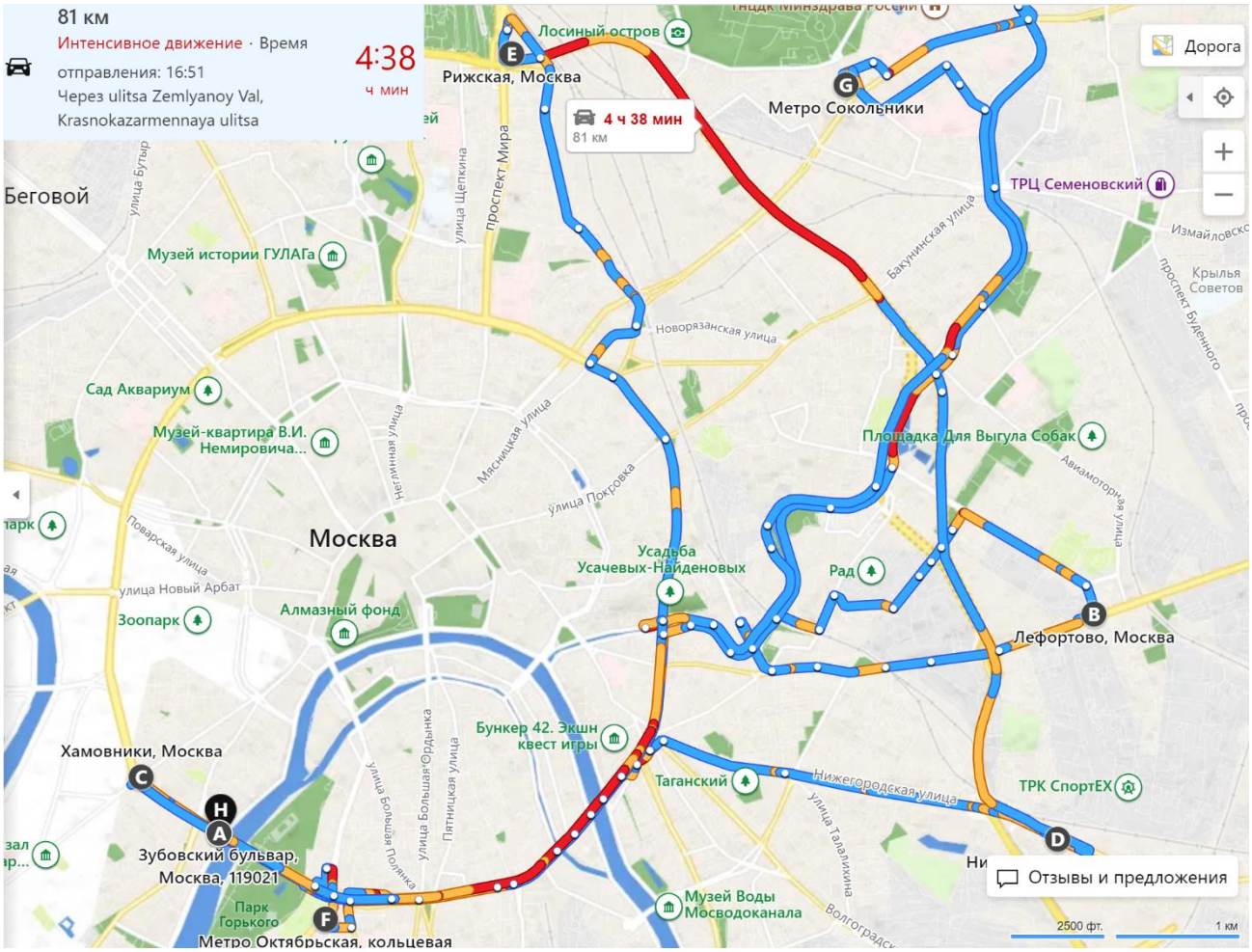

Fig. 1. Taxi route for New Year's visits

The blue-yellow-red curved lines in Figure 1 show the planned movement of a taxi for a distance of 81 km in 4 hours 38 minutes at an average speed of 17.5 kph (see the upper left corner of Figure 1). If you decide just to move around the city on foot (see the circles in Figure 2), then the distance decreases to 62 km, but it will take 12 hours 19 minutes at a speed of 5 kph.

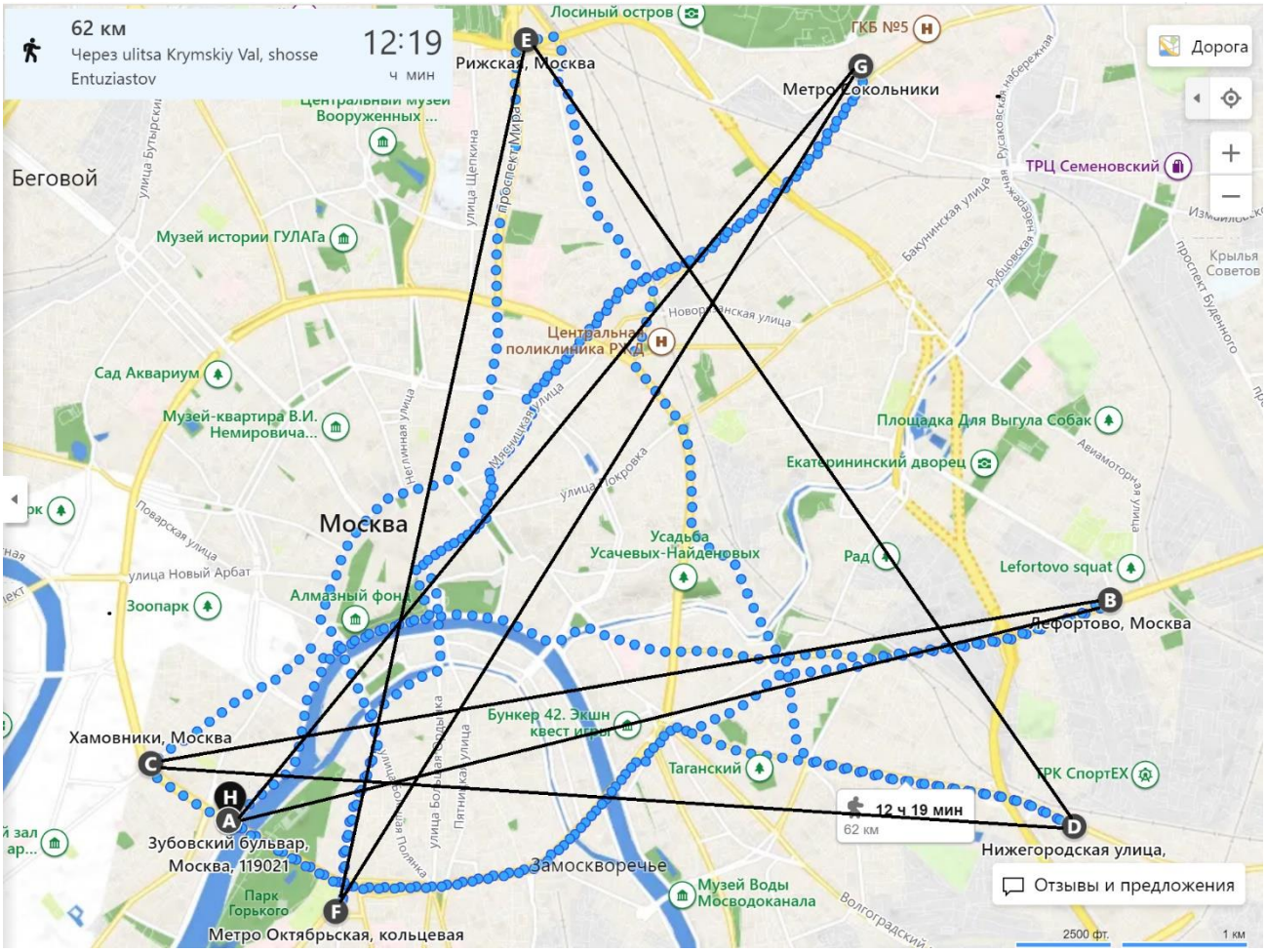

Fig. 2. Walking and helicopter route for New Year's visits

There has been talk for a long time that an air taxi is about to appear in Moscow, which will avoid traffic jams (see the red sections of the lines in Figure 1) and which will fly in straight lines (see the black straight lines in Figure 2). No doubt the hero of Chekhov's story would have used this method of transportation if he had lived to see it. And here is another not too fantastic a scenario. Suppose the hero of the story were to attach his business cards to a quadrocopter, which also flies to friends and acquaintances in straight lines, avoiding the traffic jams, dropping New Year messages, or a small gift, to them. In old Russia, on holidays, officials would come to the boss's house and sign the doorman's[2] special sheet. People who paid a visit to their peers, but did not find the owner of the house, left their business cards in the hallway. This, by the way, is where the name of these small white cardboard rectangles comes from. Here's what you can read from Gogol in "Dead Souls": "*A business card, even written even on a deuce of clubs or an ace of diamonds, was very important. Because of it, two ladies, great friends and even relatives, quarreled, precisely because one of them somehow skimped on a return visit. No matter how hard the husbands and relatives later tried to reconcile them, it turned out that though everything can be done in the world, only one thing is impossible: to reconcile two ladies who quarreled over a visit.*"[3]

[2] In Chekhov's story "The Secret", the unknown person's signature on the doorman's sheet had very interesting and comical consequences. Reader, read this story too!

[3] The extensive quoting of the classics of Russian literature can be commented on with another quote from Chekhov's vaudeville "Wedding" [8]: "They want to show their education and always talk about the incomprehensible." But the

It is immediately clear that the taxi routes (Figure 1) and walking (Figure 2) are far from optimal. One can, of course, assume that the time of the visits was agreed in advance, which would explain such long journeys from one end of the city to the other. But most likely it was like this: A husband with a New Year's hangover (see Chekhov's story) received from his wife a list of those who need to be visited. The husband, without much thought, stared blankly at the note and gave the necessary orders to the cabbies.[4]. The story, by the way, ends with a description of another scandal that the wife accused her husband of, because of the overspending of money on cabbies (5 rubles 80 kopecks).

And here is how the hero of the novel in L.N. Tolstoy's "Resurrection" [3] performed: "*Having figured out where to go first, where to go after, so as not to return, Nekhlyudov first of all went to the Senate*." Nekhlyudov in St. Petersburg did not engage in empty visits—he visited the thresholds of state institutions in the hope of changing the sentence of Katyusha Maslova—a victim of both the lust and meanness of Nekhlyudov himself, and a miscarriage of justice. Nekhlyudov, unlike the hero of Chekhov's story, before his "tour", tried to solve the traveling salesman's problem somehow in a rough outline. This problem, by the way, in our time is solved by delivery people on bicycles who carry food in yellow square bags on their shoulders. Time is money!

Let's write a small program that will solve the problem of a traveling salesman (TSP—from the English travelling salesman problem) fixed in Chekhov's story. What computer tool to use for this !? The author is used to working with Mathcad [4], but…

While working on this article, something happened to Russia that happens to it with some fatal frequency at the beginning of every century. Let's not delve too far into history, but ... The beginning of the 17th century—the Time of Troubles and the Polish intervention, the beginning of the 18th century—the Northern War with its key Battle of Poltava, the beginning of the 19th century—the invasion of Napoleon, the beginning of the 20th century—the First World War and the Civil War, the beginning the 21st century is the Ukrainian crisis, which is still unknown when and how it will end. But even in the middle of the centuries, not everything was calm either: the 19th century—the Crimean War, the 20th century—the Great Patriotic War ... The historical pendulum swings in one direction or the other (from war to peace) with a period of about half a century. We can study the war with Napoleon and the Crimean War from the works of Leo Tolstoy we mentioned. The history of relations between Russia and the West at the beginning of the 21st century is waiting for its Tolstoy. How would you like the title of the future novel: "Hybrid Warfare and the Digital World"?

Recent events, in particular the sanctions imposed by the West, make it impossible, in Russia, to download the full version of Mathcad from ptc.com and work with it for a month. After the end of the trial month, the program becomes truncated (Mathcad Express), unless a license for the full version is purchased. But now, because of the sanctions, we can't download anything and it's impossible to pay for something. Russia is suffering from Russophobia[5]. Yes, and the basis of all the basics—the Windows operating system—also hangs in the balance for us.

But, fortunately, the American program, Mathcad, has a Russian analogue called SMath (www.smath.com), which can be successfully applied to solve many mathematical and engineering problems replacing the need

---

author simply cares about the humanization of engineering education ... Well, of course, I also want to show my erudition. And not only in SMath and Mathcad, but also in the literature.

[4] Now such notes are written by the wife before sending her husband to the store. If this is a huge supermarket, then the problem of route optimization in it also results in the traveling salesman problem, more precisely, in the problem of an itinerant buyer, and not an itinerant seller.

[5] Russophobia is a chronic disease of Europe with periods of recession and exacerbation. True, in the West this disease is called differently and they believe that it is not the West that suffers from it, but Russia. But the truth, as always, lies in the middle. In the works of Tolstoy and Chekhov, a balanced assessment of this phenomenon, stretched for centuries, can be traced.

to use a foreign import. Another plus is that the SMath program works not only under Windows (like Mathcad), but also under alternative free operating systems.

So, let's download SMath, install it on a computer (this is done in a couple of minutes) and solve Chekhov's New Year's visitor problem in the simplest way, namely the nearest neighbor method, the algorithm of which belongs to the group of greedy algorithms [5]. Along the way, we note the differences between SMath and Mathcad. Or rather, not in passing, but mainly!

The scheme of the center of Moscow[6] in Figure 1 or 2 was uploaded to the Paint editor in order, firstly, to draw straight lines with arrows marking the route of an air taxi or quadrocopter over Moscow[7], and, secondly, to digitize the map—to obtain the coordinates, of the points scheduled for visits, in pixels. To do this, the cursor is brought to the each point in turn, the coordinates of which are written in the lower left corner of Paint. These coordinates are transferred to the X and Y vectors (see Figure 3). A constant was subtracted from the values of the elements of the Y vector—the number 1456. This is the height in pixels of our two pictures placed in the Paint environment for digitization. The fact is that in Paint, the coordinate origin is in the upper left corner of the picture, and in our future charts it is in the lower left corner..

Figure 3 shows the beginning of the calculation—entering the number of the point from which the visitor's route will begin (later we will generalize the task somewhat and assume that the trip can start from any point, and there can be any number of points). The start index of the variable, i, is entered through the dot i.start—just like in the good old version[8] of Mathcad (15 and below). The font of the variable changes from direct to italic automatically, which emphasizes that this is not a built-in, but a user-defined object. In Mathcad Prime, when entering a text index (the part of the variable name shifted down), the separating point was abandoned. There, the index is entered by pressing a special button labeled a2. Pressing this button again stops index entry. This allows you to have in the calculation such exotic variables as H_2O, H_2SO_4, etc. It is desirable to have such an opportunity in SMath.

We note right away that in the SMath environment there is no Mathcad's ORIGIN system variable, which sets the initial number of elements in the vector, rows and columns in the matrix, and which by default (factory settings) is zero. In the SMath environment, arrays are numbered only from one. Whether this is good or bad is a different story. In our calculation, the istart variable is assigned one—the visitor starts his trips from the first point—from the house on Zubovsky Boulevard. Then we change the value of istart from one to another value and see what happens.

[6] In Chekhov's time, it was virtually all of Moscow.

[7] We note in passing that the flight along some straight lines in Fig. 2 passes over the Kremlin. And this is unacceptable for obvious reasons. Therefore, these lines will have to be slightly bent.

[8] Many users, for a number of reasons, do not switch to the new version of Mathcad, i.e. Mathcad Prime, due to its unusual interface and the lack of some features of the old version.

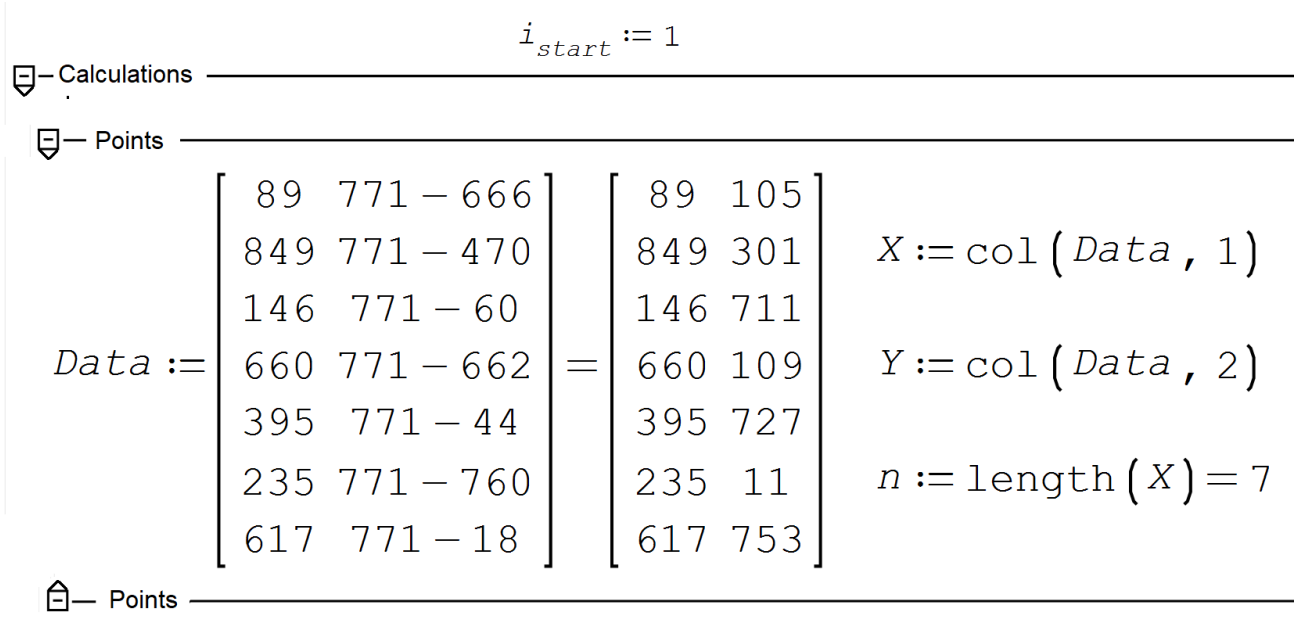

Fig. 3. Starting the calculation of the path of the traveling salesman—Chekhov's visitor

So, at the very beginning of the calculation, we press the i key in lowercase and get what is shown in Figure 3.1. The SMath package lists all built-in and user-defined constructs whose names begin with this letter in the drop-down list. This is very convenient and Mathcad does not have it. This option can be disabled, if so desired, through the menu command (switch) "View / Dynamic Input Assistance".

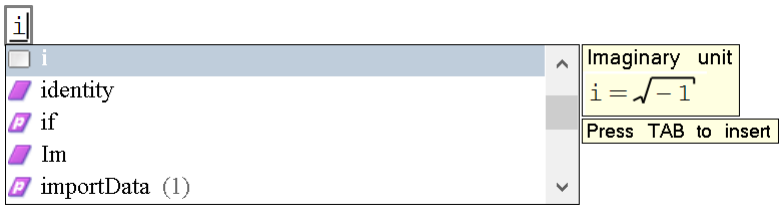

Fig. 3.1. Beginning of a list of variables and functions whose names begin with the letter i

When creating a document, it is recommended to immediately work with the so-called **areas**, where a part of a calculation may be placed. Such an area can be collapsed so that you can simultaneously see parts of the document that are far apart from each other—entering the initial data and the answer, for example. Without areas, you would have to turn the mouse wheel for this, which can be inconvenient for large numbers of calculations. And so clicking on the square with a minus (see figure 3) the area collapses. The minus turns into a plus and, before your eyes, just what you need—the initial data and the answer—appears. Click again on the plus sign, and the area opens up, showing what is inside it. Areas can be given a name and, if necessary, can be password-protected—protected from prying eyes[9]. In SMath, you can insert another, nested area inside an area, which allows you to structure the calculation. This is not available in Mathcad. Below, Figure 7 shows a "main" area, named Calculation, with four named areas nested in it, where user-defined functions are stored, which will be discussed later.

The author dreamt a little about his drawings. He drew pluses and minuses with pointers in squares not only at the beginning of the regions, but also at their ends—the way it is done in Mathcad 15 and which is not in

[9] On the Mathcad and SMath user forums, people who have forgotten their password, often ask how to open a password-protected area.

either Mathcad Prime or SMath. In addition, the nested areas were shifted to the right by the author for visual fixation of the calculation structure. Something similar is provided in the Maple package. The author informed the developer SMath about this fantasy (about a wish), who promised to implement such a convenient feature in the next version of the package.

In the first two statements of the nested area named Points in Figure 3, vectors with seven elements are assigned to the variables X and Y. This is done in the same way as in Mathcad 15—by pressing the button with the image of a square matrix of two by two elements on the **Matrix** panel, shown in Figure 3.2 below (this panel is located in the upper right corner of the SMath working window). After that, in the dialog box that appears, a request is made for the dimensions of the matrix—the number of rows and columns. But, alas, in this window there are no buttons for deleting rows and columns or for adding them. Of course, this defect needs to be corrected, and it is even better to do it as it is provided in Mathcad Prime, where it is possible to select the size of the future matrix or vector by dragging the mouse (approximately as it is done in Word and Excel) and there are buttons for deleting unnecessary elements and inserting additional elements. Rows and columns of the SMath matrix have to be deleted in rather tricky ways.

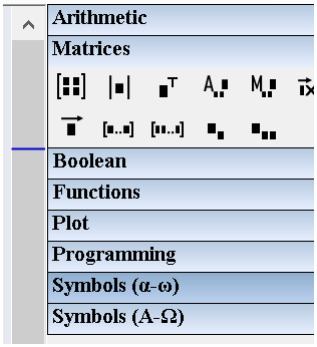

Fig. 3.2. Matrix operator panel

Note that the SMath package also allows you to download data from a file on disk. And in general, there are programs that, when you click on the picture with the mouse, will write the coordinates of the points to disk as a text file. Such a series of pairs of numbers can be automatically transferred to Smath in order to work further with the data.

Figure 4 shows one of the expanded regions with a user function that returns a square matrix containing the distances between points on the traveling salesman's path. Once upon a time a similar matrix (tablet) could be seen in country buses. The passenger entering the bus informed the conductor where he was going. The conductor looked at the matrix and notified the passenger of the fare. There were dashes along the diagonal of such a bus matrix. In SMath we store the value of infinity there, in order to prevent the salesman from going directly from one point to itself!

In the program in Figure 4, three programming structures can be seen: program blocks, local variables, and program control constructs. Program blocks (begin-end constructs in the Pascal language) in the SMath environment are fixed by a vertical bar entered using the line button (Figure 4.1). Variables automatically become local (visible only in the program) if they are entered in the body of the program—to the right of the vertical bar. We have three such variables i, j and M. Note that in SMath, the colon and equals operator (:=) is used to enter both global and local variables. You only need to press the "colon" key, and the “equals” sign will be added automatically. Better yet, press the equals key rather than the colon key. If the variable is free, then the "equals" sign will turn into a "colon and equals" sign, and the user will be given the opportunity to enter the desired value into the variable. If the variable already stores something, then pressing the "equal" key will lead to the fact that this something will be displayed. Mathcad uses a different operator for local

assignment, the left arrow, which is inconvenient and devoid of any logic. The content of the completed distance matrix (rounded to integers) is shown at the top of Figure 6.1.

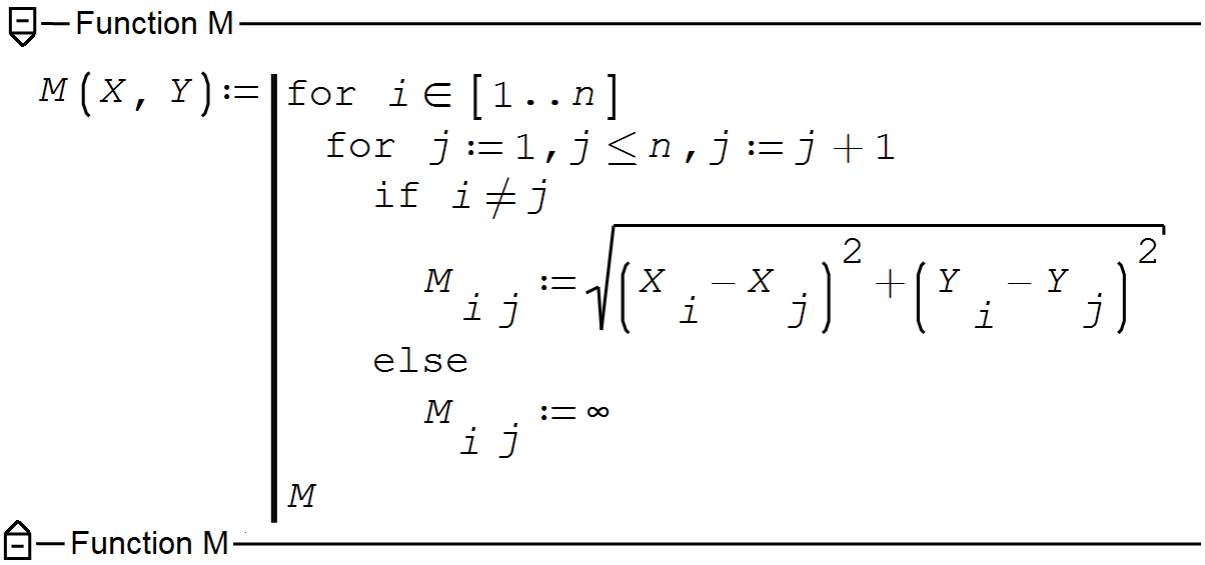

Fig. 4. A function named M that returns a square matrix of distances

The set of program control structures that change the natural order of execution of operators (from left to right and top to bottom) in the SMath environment is approximately the same (Figure 4.1) as in Mathcad. Only the return statement is missing. The on error operator (error handling) in the SMath environment has the name try (attempt). The same renaming applies in the Mathcad Prime environment..

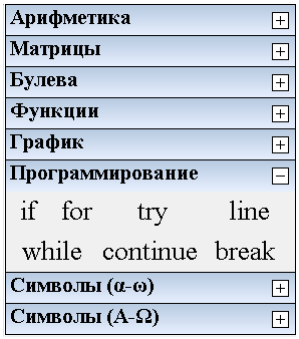

Fig. 4.1. Operator Panel Programming

The loop with the **for** parameter is implemented in two versions in SMath—with three and four operands—see Figure 4.2.

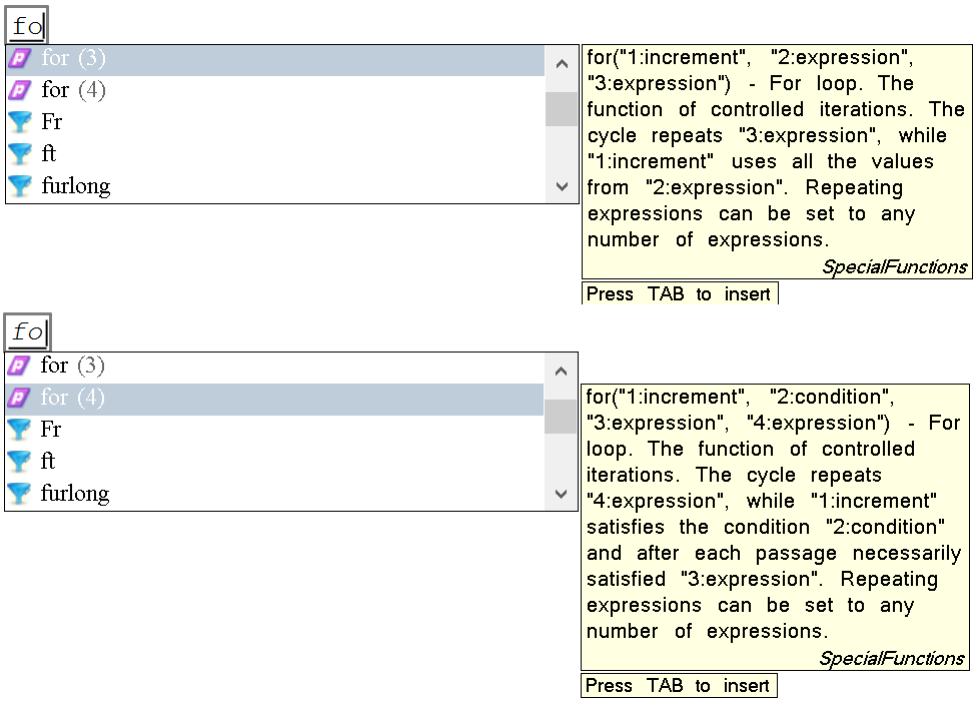

Fig. 4.2. Two forms of the for statement

In the program in Figure 4, both variants of the **for** loop have been used, although it would have been possible to limit ourselves just to the first version with three parameters. But there are cases where the second option is preferable. This occurs when programs written in C are translated into SMath, for example[10], where the four-argument form of the **for** loop is used. In addition, this form makes it legal to change the value of the loop parameter in the body of the **for** loop.

The first (outer) **for** loop contains the name of the loop parameter and the "belongs to" sign. Then so-called range variables (Range) are usually inserted here, using an arithmetic progression. The blanks of this progression are in the **Matrix** panel (Figure 3.2) in two versions—with the second value of the range variable [1.1, 1.2 .. n] and without it [1 .. n] as in Figure 4. In the second case, when no step is specified, the second value of the range variable is one greater than the first (the step will be one). Or one smaller than the first (minus one) if the first value of the range variable is less than the last.

We emphasize right away that the range variable in the SMath environment is a full-fledged vector, to which all vector operations are applicable—see examples in Figure 4.3. Unfortunately, this is not available in Mathcad. There, if necessary, you will have to do special complex procedures to convert the Range variable into a full-fledged vector.

$$V := [2; 1{,}5 .. 0] = \begin{bmatrix} 2 \\ 1{,}5 \\ 1 \\ 0{,}5 \\ 0 \end{bmatrix} \qquad V_2 = 1{,}5 \qquad \text{reverse}(V) = \begin{bmatrix} 0 \\ 0{,}5 \\ 1 \\ 1{,}5 \\ 2 \end{bmatrix} \qquad \begin{matrix} \min(V) = 0 \\ \\ \max(V) = 2 \end{matrix}$$

[10] We invite the reader to find on the Internet and implement in SMath a C program that solves the traveling salesman problem using more advanced methods—the annealing method, ant colony, genetic, etc.

Fig. 4.3. Examples of working with a range variable as a full-fledged vector

And here is another useful property of SMath, which, frankly, should have been mentioned at the very beginning. From Figure 4.3 it can be seen that in decimal numbers a comma is used, not a dot, and the argument separator in functions is not a comma, but a semicolon. The point, comma and other "punctuation marks" in the calculations can be selected by calling the Options dialog box from the Tools menu, shown in Figure 4.4. Here the Russian version of Excel immediately comes to mind, with a comma in numbers and a semicolon in lists. A comma in the numbers of the program is convenient when preparing articles, when Russian editors require the use of a comma, not a dot in non-integer numbers.

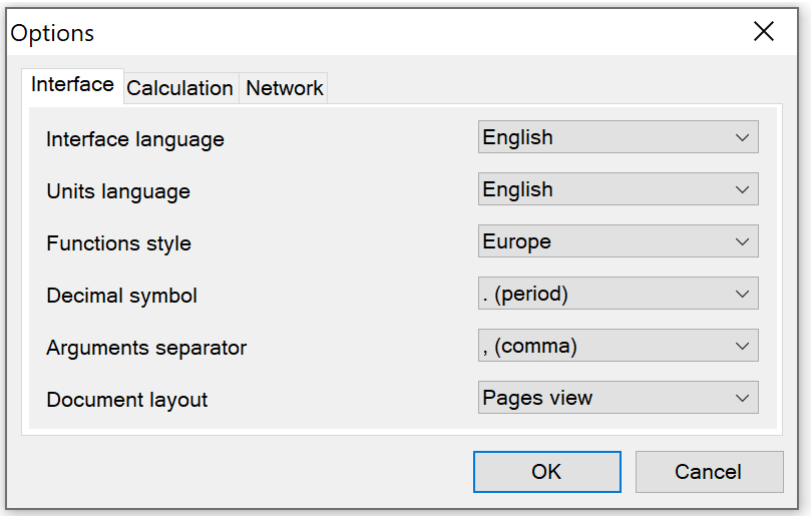

Fig. 4.4. Menu Options

Figure 5 shows a user function that returns the index of the minimum element of the vector. A simple enumeration of vector elements is used. In Mathcad, the search for the desired indices in arrays (in vectors and matrices) is carried out by the built-in match function with two arguments, which returns the desired arrays of index values. Our function in Figure 5 returns a scalar, even if the argument vector contains multiple minimum values. This is enough for us, but the match function and others like it would be useful in the SMath environment as well..

In Figure 5, the function is called immediately after creating a function to check its performance. It is shown that in the vector X (the abscissas of the travel points of the Chekhov visitor—see Figure 1-3) the minimum element 212 (the built-in function min is involved here) is in third place. If there were more than one such element, then the built-in Mathcad **match** function would return a vector. In our case, the $\mathbf{i}_{min}$ function in such a situation will return a scalar—the last value in the series of minimum values.

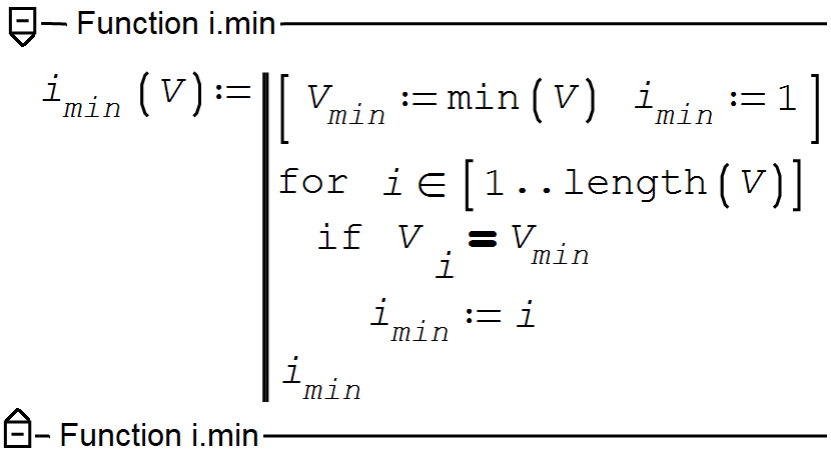

Fig. 5. A function that returns the index of the minimum element of the vector

So, the auxiliary functions have been created and the main function can be introduced into the calculation—a function that returns a vector containing the numbers of points through which the salesman must go, guided by the nearest neighbor algorithm—see Figure 6. It is quite simple and does not require much explanation. It has two **for** loops, the second of which is nested within the first (see also Figure 4). In the nested loop, a vector S is formed that stores the distances from the current j-th point to the rest of the route points. The index of the minimum element of this vector, found using the $\mathbf{i_{min}}$ function, becomes the number of the next point on the traveling salesman's path after the execution of the statement $Way_i := \mathbf{i_{min}}(S)$. After that, the corresponding elements of the distance matrix from the i-th to the j-th point are assigned the value of infinity so that the salesman does not walk along this route one more time. On Figure 6.1 you can see what is contained in the distance matrix M before and after the implementation of the nearest neighbor algorithm.

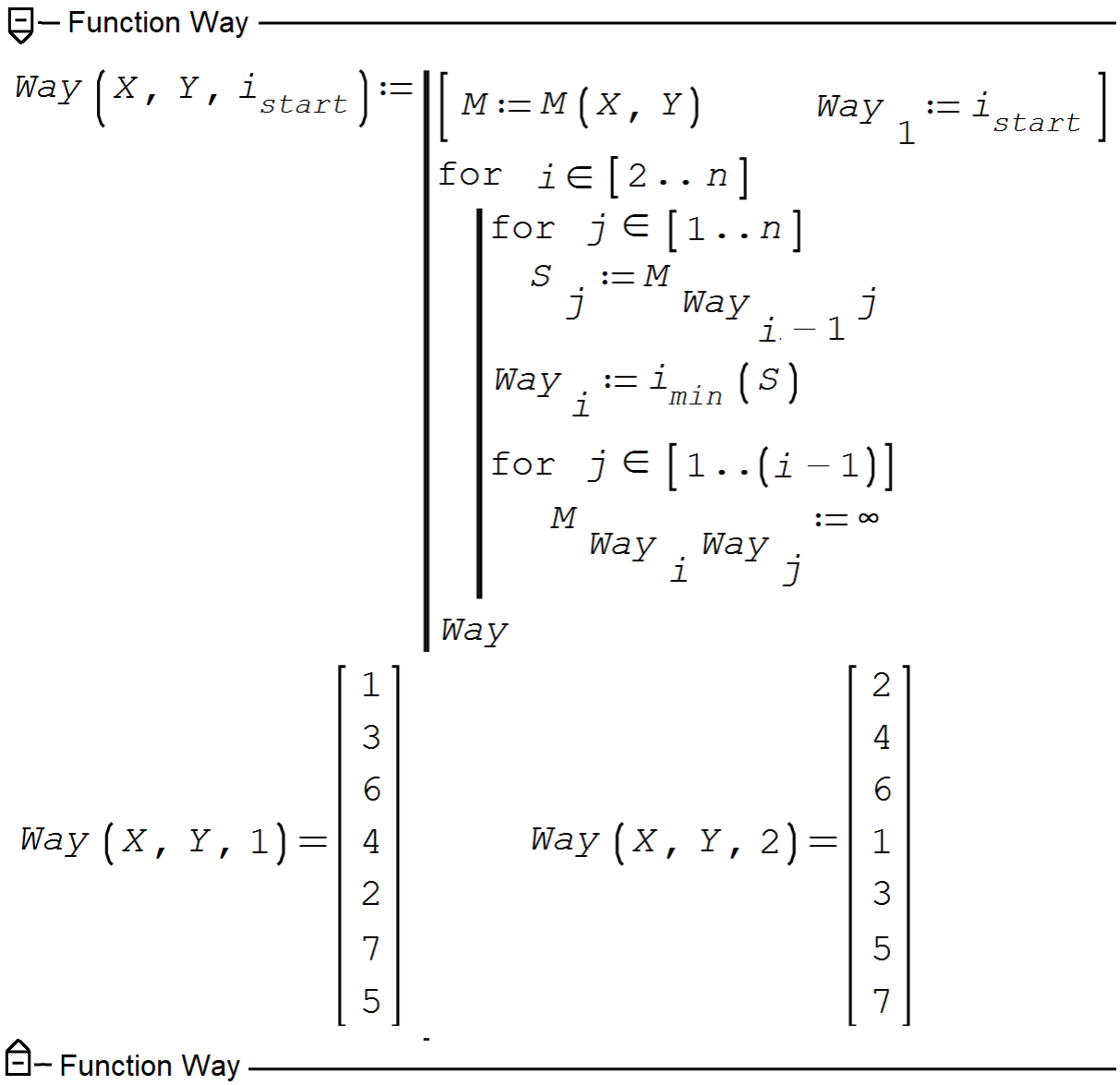

Fig. 6. Program for solving the traveling salesman problem by the nearest neighbor method

Figure 6 also shows examples of calling the *Way* function for two different starting points (see also Figure 8). In the program in Figure 6, if the word, *Way,* at the end of the program is replaced by the letter M, then the matrix will be returned with the original "infinities" along the main diagonal and with the added "infinities" after the implementation of the nearest neighbor method.

$$\begin{bmatrix} \infty & 1386 & 130 & 1291 & 1283 & 216 & 1498 \\ 1386 & \infty & 1495 & 341 & 1256 & 1271 & 910 \\ 130 & 1495 & \infty & 1414 & 1289 & 341 & 1546 \\ 1291 & 341 & 1414 & \infty & 1477 & 1134 & 1206 \\ 1283 & 1256 & 1289 & 1477 & \infty & 1372 & 516 \\ 216 & 1271 & 341 & 1134 & 1372 & \infty & 1515 \\ 1498 & 910 & 1546 & 1206 & 516 & 1515 & \infty \end{bmatrix}$$

$$\begin{bmatrix} \infty & 1386 & 130 & 1291 & 1283 & 216 & 1498 \\ \infty & \infty & \infty & \infty & 1256 & \infty & 910 \\ \infty & 1495 & \infty & 1414 & 1289 & 341 & 1546 \\ \infty & 341 & \infty & \infty & 1477 & \infty & 1206 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 1271 & \infty & 1134 & 1372 & \infty & 1515 \\ \infty & \infty & \infty & \infty & 516 & \infty & \infty \end{bmatrix}$$

Fig. 6.1. The contents of the distance matrix before (top) and after (bottom) the implementation of the nearest neighbor algorithm

In Figure 3.2, on the **Matrix** panel, two buttons are visible for accessing the elements of vectors and matrices. In the Mathcad environment, one button is intended for this, and matrix indices are separated from each other not by a space, but by a comma. The function in Figure 6 is a small masterpiece in terms of working with indexes—textual (imin, istart,) and numerics—with operators that return an element of a vector or matrix. Some of our operators are "three-level", when, for example, the first index of the matrix M is the index of the *Way* vector.

Figure 7 shows the complete calculation with four nested collapsed regions storing point data and the three user-defined functions described above. It remains only to build a graph and animate it. To do this, we use the operators shown at the bottom of Figure 7. Be warned that the technology for plotting graphs – both 2D and 3D—is fundamentally different in SMath from that in Mathcad.

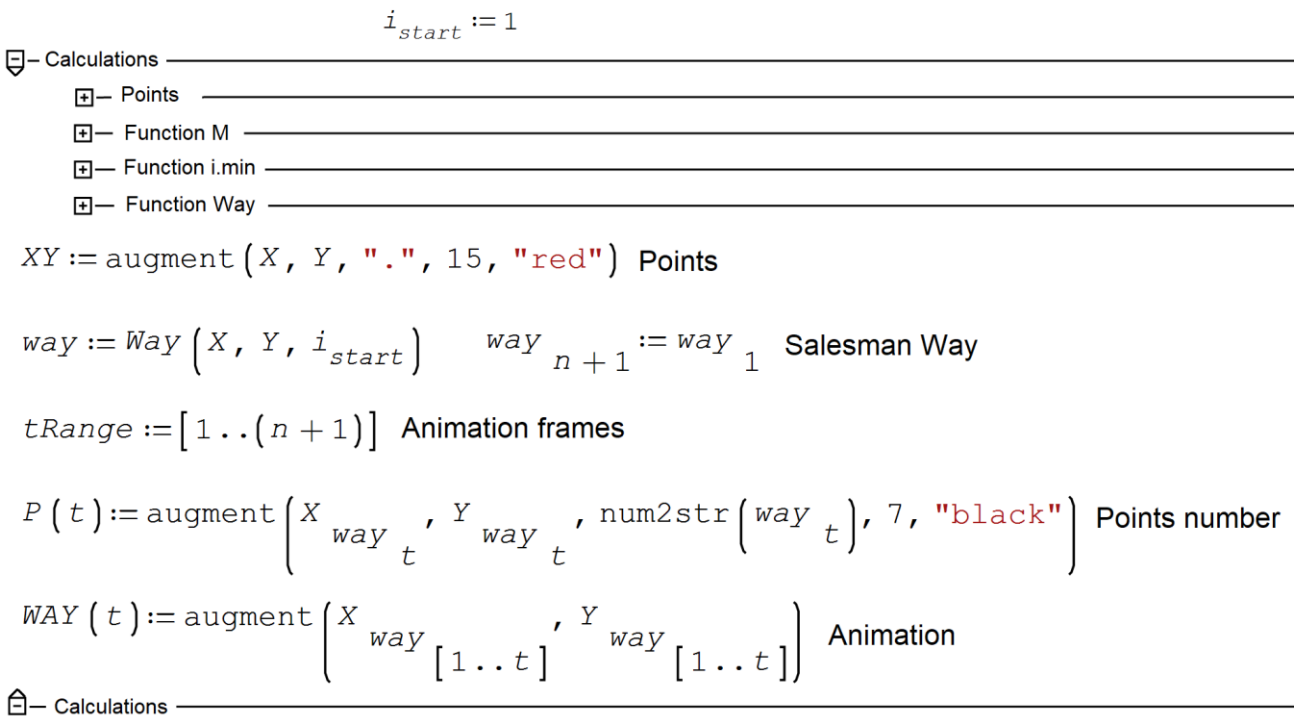

Fig. 7. Operators needed to create a graph and animation of the traveling salesman

In Figure 7, an XY matrix is formed with n rows and with the following five columns: vector X, vector Y, points (".") with a size of 15 units and with red color. In addition to points, you can use crosses (letter x or "+"), circles (letter o) and asterisks ("*"). Other punctuation marks and letters (Russian and Latin) are written a little to the right and below the specified place on the chart. This is very convenient, and Mathcad does not have it.We will use this when we assign the number of the point through which the traveling salesman passes (the penultimate operator in Figure 7).

In Mathcad, animation is controlled by the FRAME system variable. In the SMath environment, this work is done by the variable t, which must be made an argument of the animated objects. We have *P(t)* and *WAY(t)*. Animation frame numbers must be stored in a vector or range variable. For us, this will be the *tRange* variable – see the dialog box in Figure 8.2. This window can be accessed by clicking the right mouse button on the activated chart. The graph is inserted into the working document through the menu commands shown in Figure 8.1. The animation frame rate and other animation parameters are set through the dialog box shown in Figure 8.3.

Figure 8 shows the paths of the Chekhov visitor from two different starting points—from the first (visitor's house on Zubovsky Boulevard) and from the second (Lefortovo—see also Figure 6). During animation (and it can be seen in point 6 here https://community.ptc.com/t5/Mathcad/Mathcad-vs-SMath/td-p/802499) a blue straight line comes out of the starting point (first or second) and “bypasses” other red dots, to the left and below which their numbers will appear.

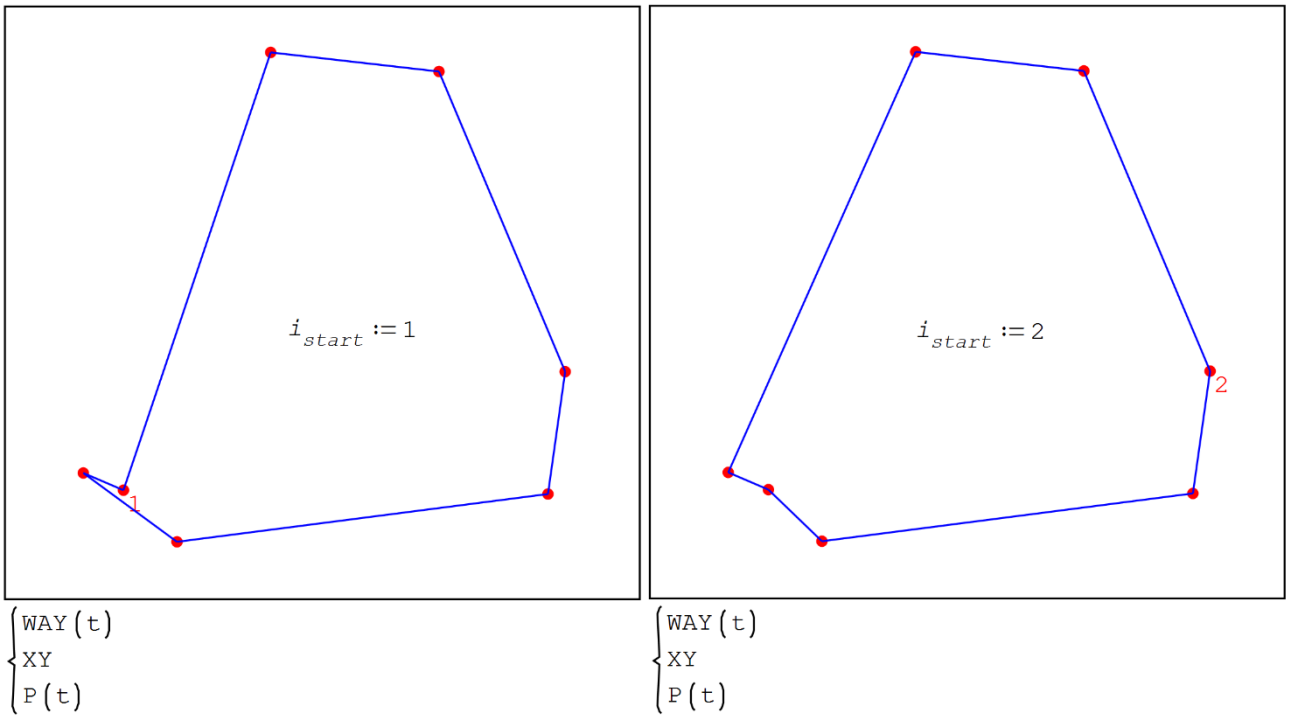

Fig. 8. The path of the visitor from Chekhov's story from two different starting points.

To minimize the length of the route, our Chekhov visitor would have had to go from Zubovsky Boulevard (point 1) not to Lefortovo (point 2), but to Khamovniki (3), then to the Kaluga Gates (6), then to the Nizhegorodsky railway station (4), and then Lefortovo (2). Further, his route would have to pass through Sokolniki (7), Krestovskaya Zastava (5) and end at home on Zubovsky Boulevard (1). But the right graph in Figure 8 shows a shorter route: in Khamovniki it was necessary not to be greedy and go not to the nearest Kaluga Gates, but to the more distant Krestovskaya Zastava. The optimal solution to the traveling salesman problem should in principle be independent of the starting point. In our case, this is not so—see Figure 8.

But all this applies to a route consisting of segments of straight lines (helicopter travel—see Figure 2). In reality, an application on a computer or smartphone would have to offer a person ordering a circular route for a taxi (see Figure 1) to optimize the trip by solving the traveling salesman problem, which takes into account not only the location of stops, but also the traffic situation, minimizing not only the distance traveled , but also the cost of the trip.

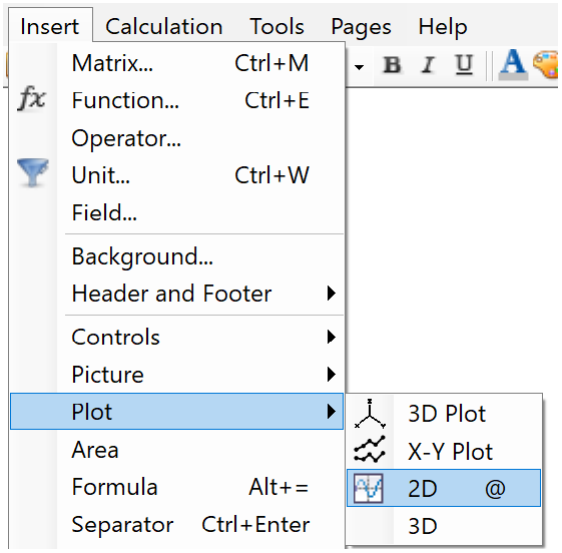

Рис. 8.1. Menu Commands for Inserting a 2D Plot

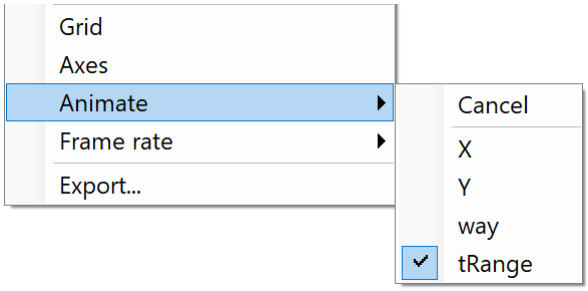

Fig. 8.2. Setting an animation variable

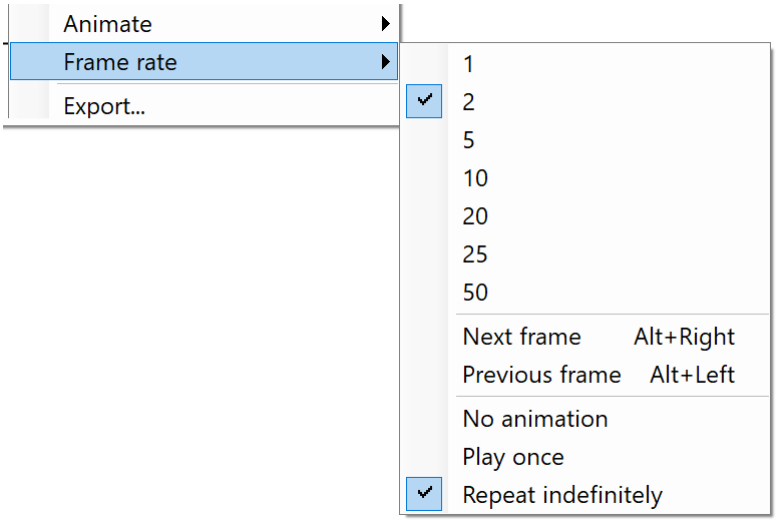

Fig. 8.3. Setting animation options

Figure 9 shows three animation frames of a traveling salesman moving over 50 random points, which can be entered into the X and Y vectors through a function that returns random numbers.

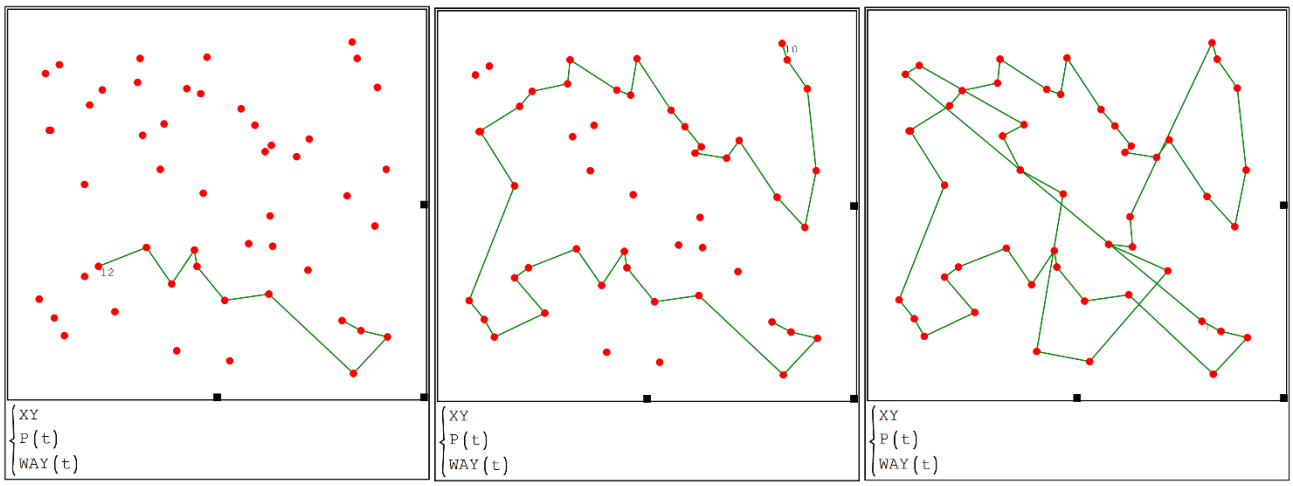

Fig. 9. Three frames of animation of the traveling salesman moving along fifty random points

If the program in Figure 6 were to replace the min function with the max function, and the infinity sign with a negative infinity, then the furthest neighbor algorithm would be implemented—see Figure 10. You could go through all the starting points and choose an even longer route than the one that the wife sent her husband to visit in Chekhov's story!

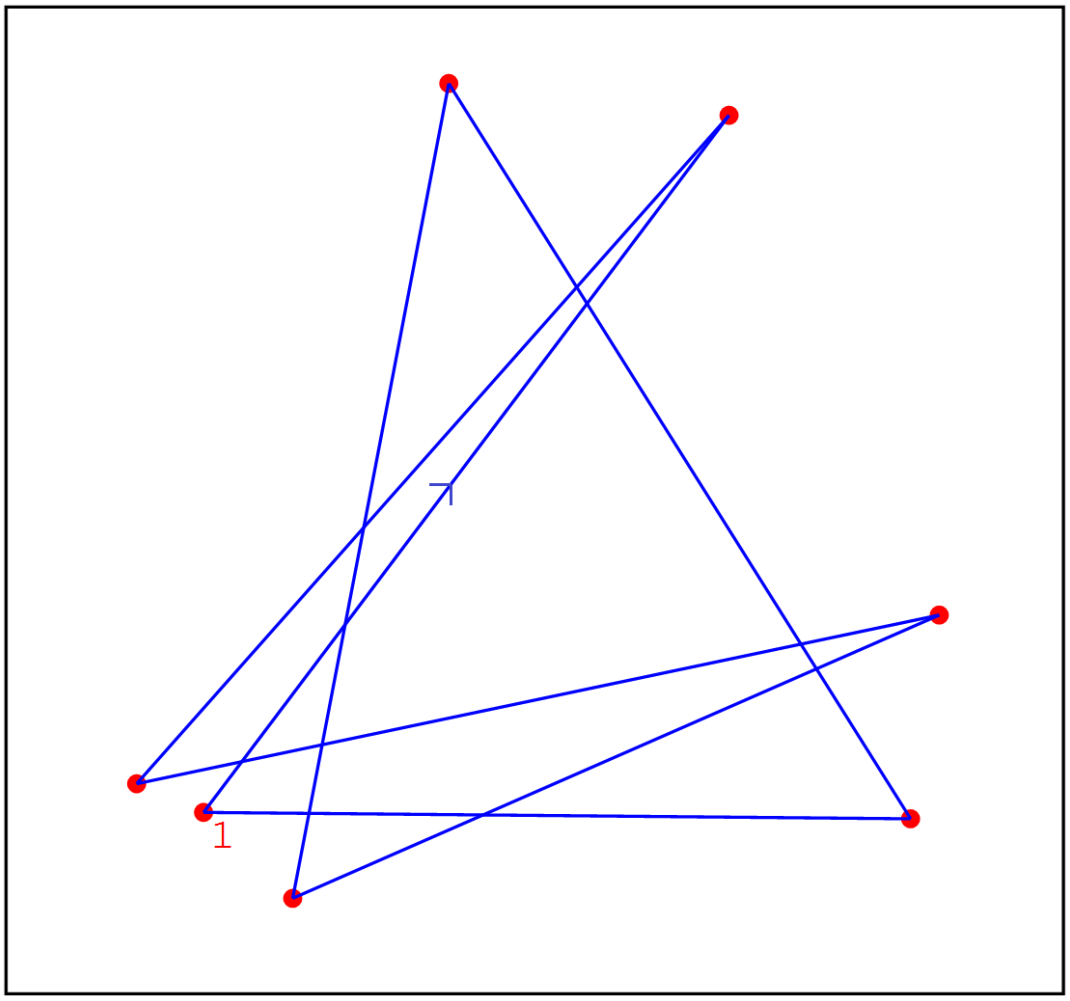

Fig. 10. The visitor's path according to the furthest neighbor algorithm

Figure 11 shows a picture from the Wikipedia article "Traveling Salesman Problem" with an explanation of what the result of trying to solve the traveling salesman problem by brute force. Frau sends her Herr to fly around on a plane with friends in 15 cities in Germany ...

Fig. 11. Solving the traveling salesman problem for 15 cities in Germany.

The site [7] contains extensive information about the traveling salesman problem. In particular, there you can find the path of a traveling merchant through all the settlements of Italy (Figure 12) and other countries, and even the whole World. Of course, more advanced algorithms are used, eliminating loops and truly minimizing the paths. But these routes have not been proven to be the shortest. We recommend that readers implement such algorithms in SMath—to create a new Way function. All such algorithms are heuristic, using practical methods that are not guaranteed to be accurate or optimal, but sufficient to solve the problem. But for a small number of points (Figure 8), you can find an absolutely accurate solution even without using complex algorithms.

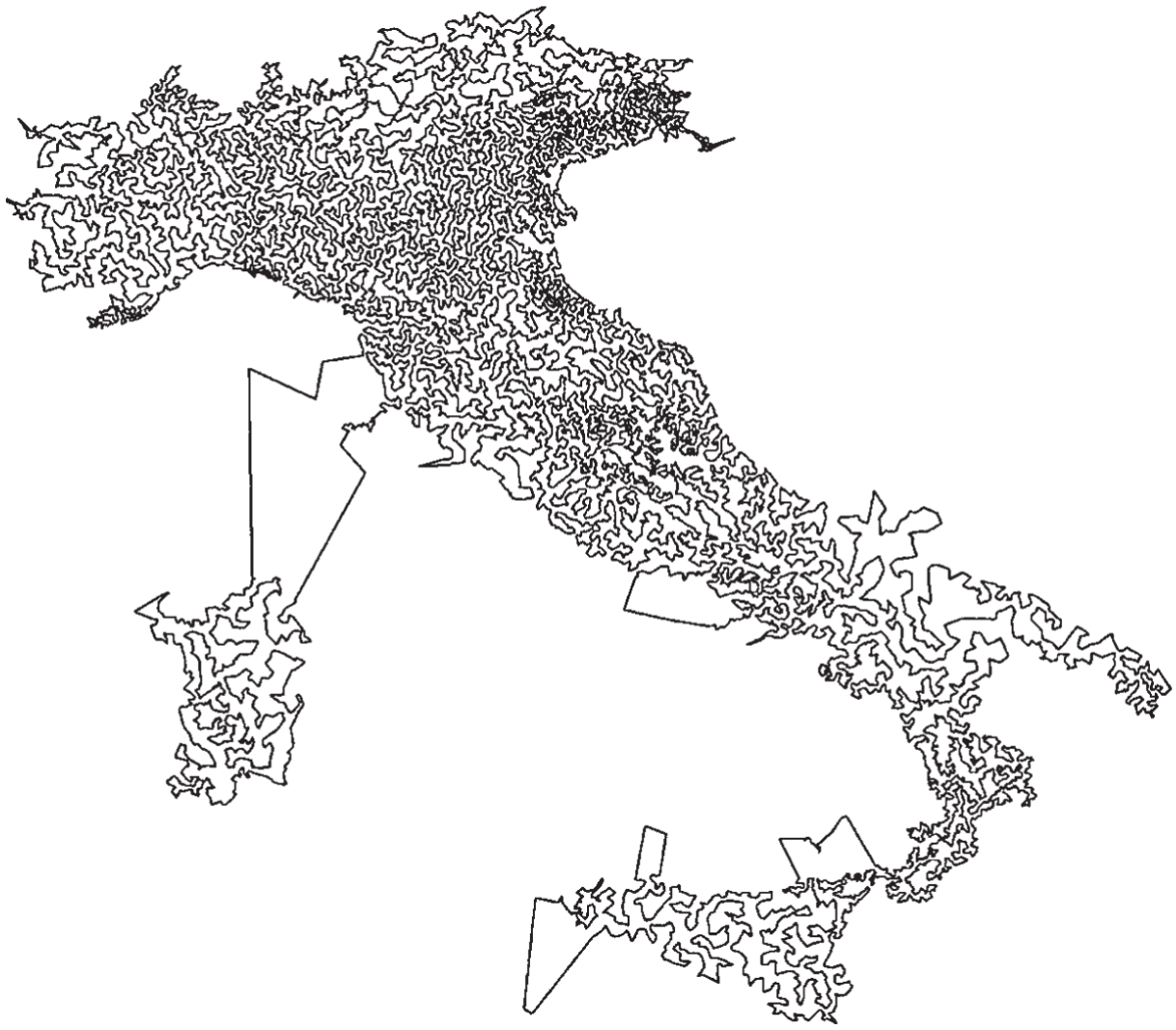

Fig. 12. Salesman travels through Italy

In February 2023, when the work on the book was almost finished, the first author of this tutorial received a message from Canada from the developer of the Maple mathematical package—from MapleSoft, that a beta version of a mathematical program with tools to solve the travelling salesman problem had appeared. The author immediately wanted to test these tools on the problem of Chekhov's visitor—see Figures 13-15. Here's what he did!

Figure 13 shows the following:

- The *with*(*GraphTheory*) command loads tools for working with graphs in the Maple environment.
- In the Points variable, the user writes a matrix with seven rows and four columns that stores information about the trip of the Chekhov visitor. To obtain this data, the trip schematic (Figure 1 or 2) should be placed, for example, in the Paint graphics editor environment, so that the cursor can be moved from point to point to read the coordinates in pixels. The Points table is actually not a table, but the simplest relativistic database with four fields (the letters marking the places in the city, their names, their abscissae and ordinates) and seven entries. The coordinates of places in the city can also be obtained in the following way—open a map of the city on the computer, place the mouse cursor at

the desired place and read its coordinates: northern latitude and eastern longitude. However, it will then be necessary to convert these angles into kilometers on a flat map. Instead of entering the points as a database one can enter them as a square matrix with an empty main diagonal that stores the distances between the points or the time (cost) of the trip between them. If an element of such a matrix $m_{i,j}$ stores the value 100, then this means that 100 is the distance between the *i*-th and *j*-th points, or the travel time, its cost, etc. Not only will the main diagonal be empty in such a square matrix, but also some other elements, which means that a direct trip between the two cities indicated is not possible. The *CompleteGraph* command informs the user that we are dealing with an undirected graph that has 7 vertices (places in Moscow) and 21 edges (paths connecting these places). An edge of a directed graph can only be "travelled" in one direction (a one-way street).

• The vertices and edges of our graph are built by the *DrawGraph* command. The result is a regular heptagon, all vertices of which are arranged in a circle (*style=circle*) and connected by straight lines. On these edge lines, you can optionally mark the distances between points, the time of the trip or its cost.

```
with(GraphTheory) :
```

$$Points := \begin{bmatrix} "A" & "Zubovsky\ Boulevard\ " & 334 & 1276 \\ "B" & "Lefortovo" & 1681 & 951 \\ "C" & "Khamovniki" & 212 & 1230 \\ "D" & "Nizhny\ Novgorod\ Station" & 1625 & 1287 \\ "E" & "Krestovskaya\ Zastava" & 782 & 74 \\ "F" & "Kaluga\ Gates" & 498 & 1416 \\ "G" & "Sokolnitskaya\ Grove" & 1295 & 126 \end{bmatrix} :$$

```
G := CompleteGraph( convert(Points[ .., 1], list), vertexpositions = Points[ .., [3, 4]])
```

$$G := Graph\ 1:\ an\ undirected\ graph\ with\ 7\ vertices\ and\ 21\ edge(s) \quad \textbf{(1)}$$

```
DrawGraph(G, style = circle)
```

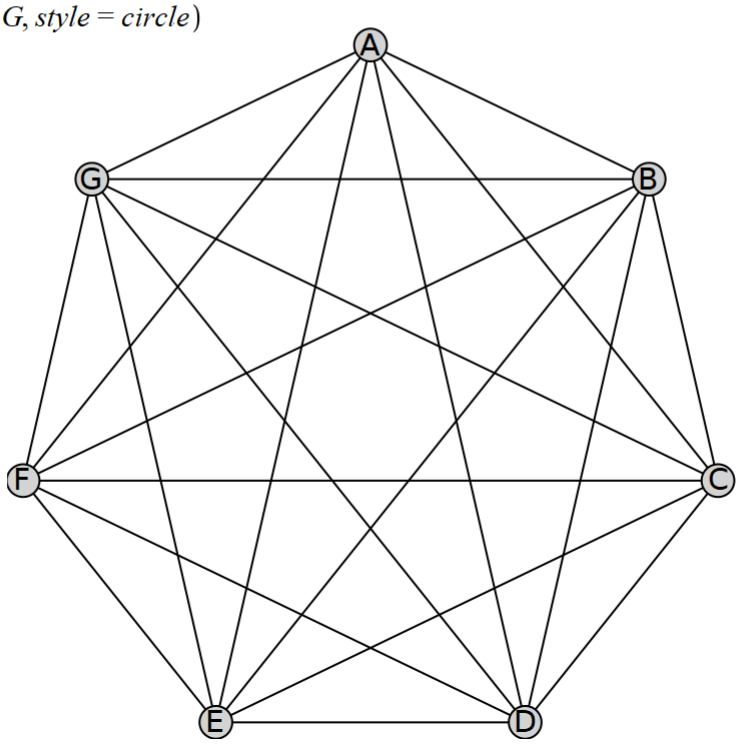

Fig. 14. Solving the traveling salesman problem in Maple 2023 (beginning)

The *TravelingSalesman* function (Figure 15) returned, firstly, the distance that the Chekhov visitor traveled (6035 pixels), and secondly, his route, following which this distance would be minimal. The other two operators in Figure 15 build the route of Chekhov's visitor, or rather, the quadcopter he launched. The points are the same as those shown in Figure 1 and 1 with only one difference—they are inverted vertically and horizontally. The south of Moscow turned out to be in the north, and the west in the east. But it's not that important. The main thing is that the problem has been solved: from Zubovsky Boulevard (point A) it was necessary to go (fly in a straight line) either to Khamovniki (C) or to the Kaluga Gates (F), and then “with all stops”. And Chekhov's visitor "travelled" as far as Lefortovo (B). The minimum route will be when departing from any point in any direction. But this last statement will turn out to be false if you can drive along some edge in only one direction.

$$W, T := TravelingSalesman(\ G, vertexpositions, startvertex = \text{"A"}\ )$$
$$W, T := 6035.03350860640, [\text{"A"}, \text{"C"}, \text{"E"}, \text{"G"}, \text{"B"}, \text{"D"}, \text{"F"}, \text{"A"}] \quad \textbf{(2)}$$

$$TG := Subgraph(\ G, Trail(T)\ )$$
$$TG := Graph\ 2:\ an\ undirected\ graph\ with\ 7\ vertices\ and\ 7\ edge(s) \quad \textbf{(3)}$$

$$DrawGraph(TG)$$

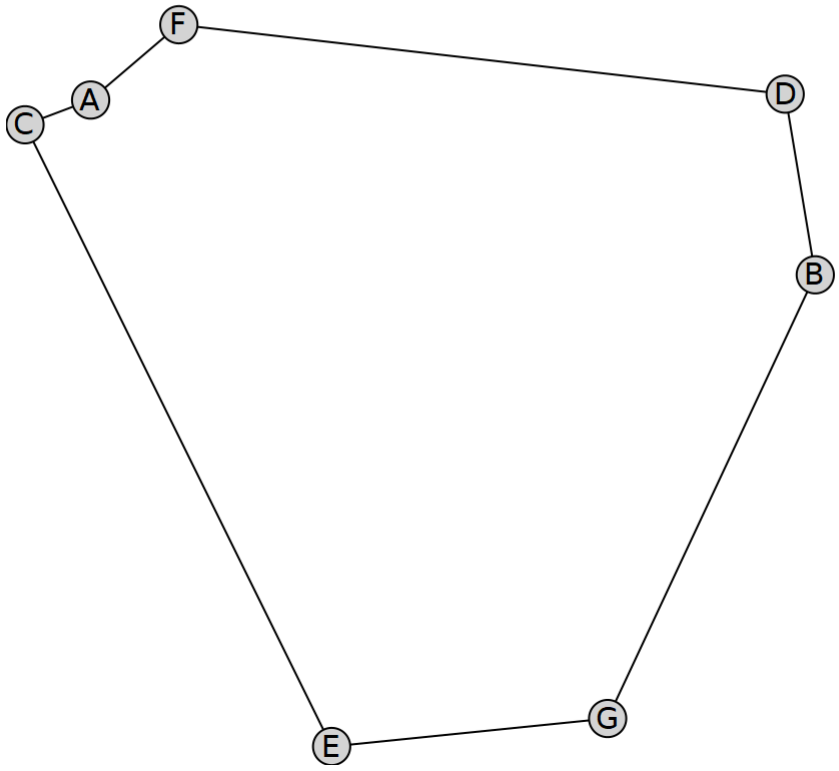

Fig. 15. Solving the traveling salesman problem in Maple 2023 (continued)

Figure 16 shows the original graph, on which the so-called Hamiltonian loop (cycle) with an optimized route is additionally drawn with red thick lines (see https://en.wikipedia.org/wiki/Hamiltonian_path).

HighlightTrail(*G*, *T*, *red*)
DrawGraph(*G*, *style* = *circle*)

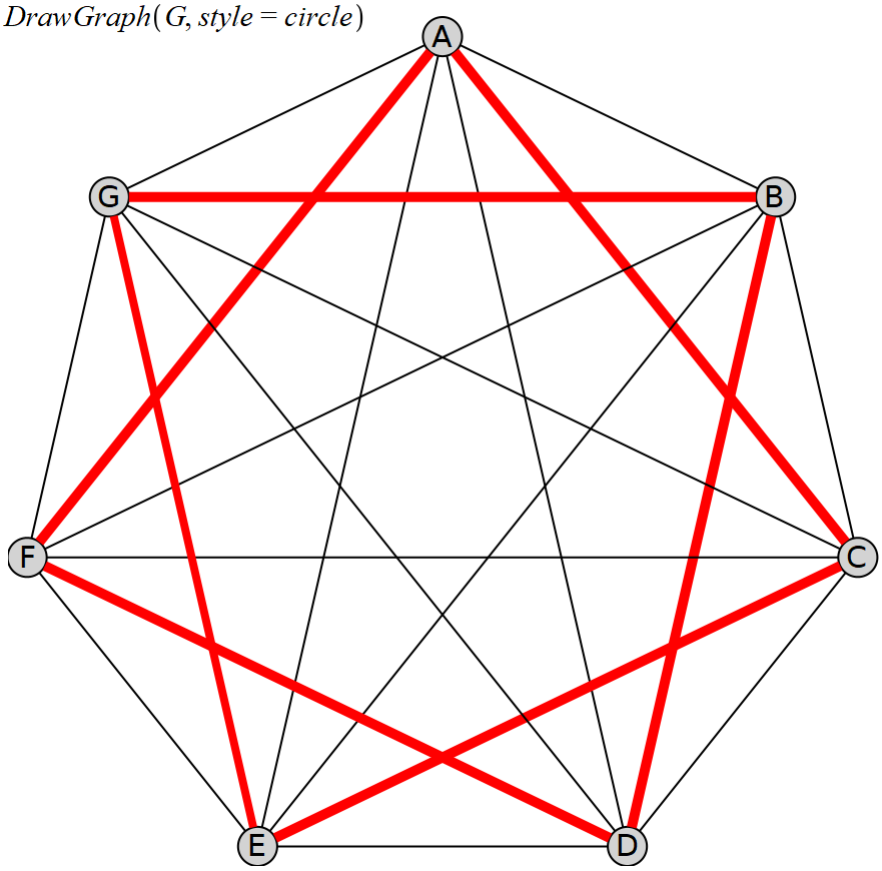

Fig. 16. Solving the traveling salesman problem in Maple 2023 (end)

Using the Maple package to solve a problem from a Chekhov story is, of course, firing a cannon at sparrows (an equivalent English phrase is: “using a sledgehammer to crack a nut”). The optimal route could be drawn by simply looking at the map of Moscow. But with an increase in the number of points, a computer can no longer be dispensed with. Here is how, for example, the traveling salesman route and its graph with 20 points might look like—see Figure 17.

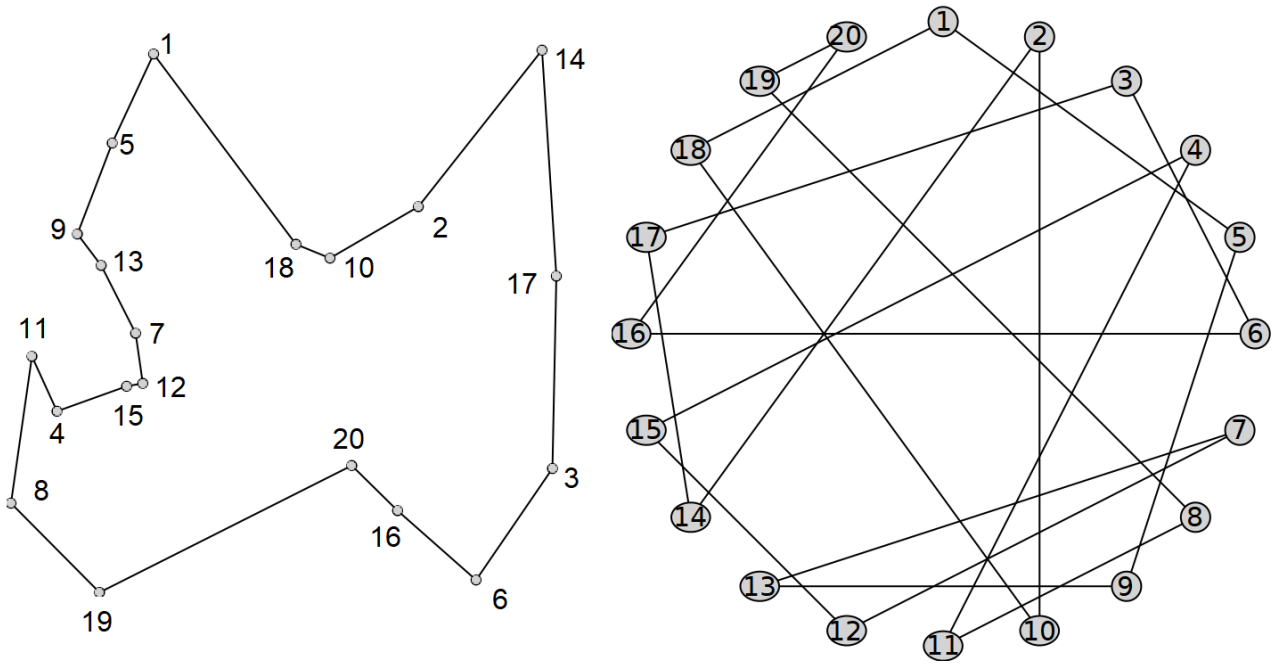

Fig. 17. Solving the traveling salesman problem in Maple 2023 (option-2)

The help of the Maple package describes a more complex task, which cannot be solved without a computer either—flying by plane along the shortest route of the 100 largest cities in Africa—see Figure 18.

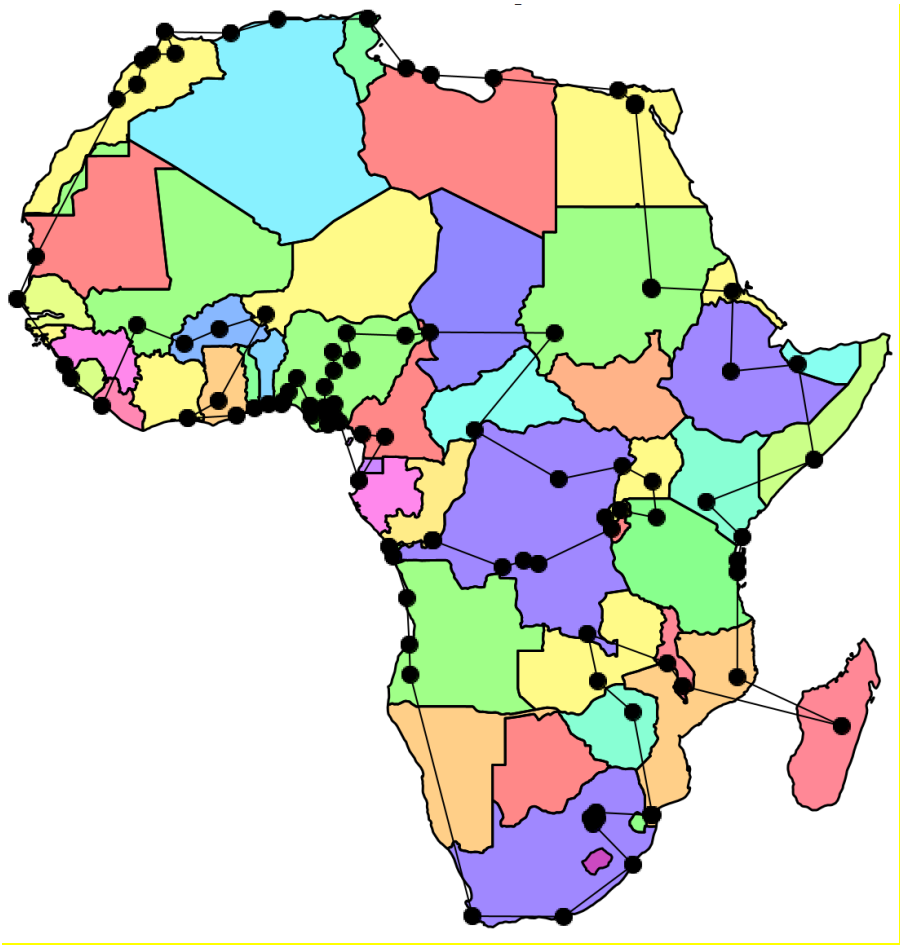

Fig. 18. Graphical display of the solution to the traveling salesman problem in the Maple 2023 environment (one hundred cities in Africa)

The map of Africa, by the way, reminds us of another interesting optimization problem indirectly related to graph theory. This is the problem of the minimum number of colors that can be used to paint the political map of the world. It has been proven that only four colors are sufficient for this, as shown in Figure 18 But in popular science magazines, similar to the one that the reader holds in his hands (opened on his computer, tablet, smartphone), contour maps regularly appear in April Fools' issues with the statement that they cannot be painted in only four colors ...

Let's return to the traveling salesman problem.

It has been proven that moving around the city with minimization of distance, time or money should be done by combining modes of transport. You need to ride a bike, periodically dismounting or rolling the bike into public transport (in a tram-bus-metro) or even in a taxi. We are now witnessing this with our own eyes, for example, in Moscow, where cyclists with yellow square bags over their shoulders are racing at breakneck speed. They deliver groceries and ready-made meals to customers, which Muscovites are “addicted to” after the pandemic. In the hands of these commercial bikers, you can see a smartphone that not only shows the way, but also optimizes the route, solving a rather difficult traveling salesman problem…

Another example. One world-famous courier service, transporting goods around the world on ships, planes, trains, cars, bicycles, etc., hired mathematicians and programmers to optimize transportation. Mathematicians with programmers created a program that allowed this service (company) to save almost two billion euros a year.

And there are many such examples.

Assignments for readers.

1. Recreate the calculation described in the article in SMath.

2. Enter 50, 100 or more random points into the calculation, select the initial one and build a traveling salesman route from it using the nearest neighbor algorithm. Then iterate over all points as starting points and find the minimum route.
3. Implement more optimal traveling salesman routes in the SMath environment by finding their algorithms on the Internet or in other sources.

Literature and links:

1. chehov-lit.ru/chehov/text/novogodnyaya-pytka.htm
2. https://knigavuhe.org/book/novogodnjaja-pytka/
3. http://az.lib.ru/t/tolstoj_lew_nikolaewich/text_0090.shtml
4. V. F. Ochkov, E. P. Bogomolova, and D. A. Ivanov, Russ. Physical and mathematical studies with Mathcad and the Internet: textbook. allowance. 2nd ed. St. Petersburg: Lan, 2018. 560 p. URL: http://twt.mpei.ac.ru/ochkov/T-2018/PhysMathStudies.pdf
5. V. F. Ochkov, A. O. Ivanova, and M. D. Alekseev, Russ. Three greedy algorithms // Informatics at school. 2018. No. 9. P. 34–42. URL: http://twt.mpei.ac.ru/ochkov/3-Problem.pdf
6. http://chehov-lit.ru/chehov/text/tajna.htm
7. http://www.math.uwaterloo.ca/tsp
8. https://ilibrary.ru/text/967/p.1/index.html