第 1 章 俄罗斯文学、旅行商问题、制裁、SMath、Maple...

本章描述了 STEM 技术在数学、计算机科学、文学、莫斯科研究甚至政治学的交叉点上的一课。 描述了一种在数学程序 SMath 的环境中通过最近邻方法解决旅行商问题的技术。

关键词：图论，旅行商问题，最近邻法，SMath，Maple，编程

契诃夫有一个相当有趣的故事：“新年的折磨”。 文本很短 — 值得阅读 [1] 或收听 [2]，然后返回本文。 故事描述了一个丑陋的妻子如何将丈夫赶出家门去拜访亲戚和朋友——祝贺他们过年：“你想不去拜访！”。 为此，有必要沿着以下路线绕莫斯科一圈：Zubovsky 大道（起点是故事主人公居住的地方）、Lefortovo 的红色军营、Khamovniki、下诺夫哥罗德车站 [这是一个临时车站-----莫斯科-下诺夫哥罗德铁路。后来延伸至库尔斯克火车站，下诺夫哥罗德火车站本身也被拆除。 从它仍然是 Nizhegorodskaya 街的名字。Krestovskaya Zastava 在现在的 Rizhskaya 地铁站区域，Kaluga Gate 在 Oktyabrskaya 地铁站区域，以前叫 Kaluzhskaya。]，Krestovskaya Zastava, Kaluga Gates, Sokolnitskaya Grove, 返回 Zubovsky 大道的所在地。

作者尝试使用一种用于预订出租车的应用程序来预订这样的循环行程。 这是他所做的——见图 1.1。

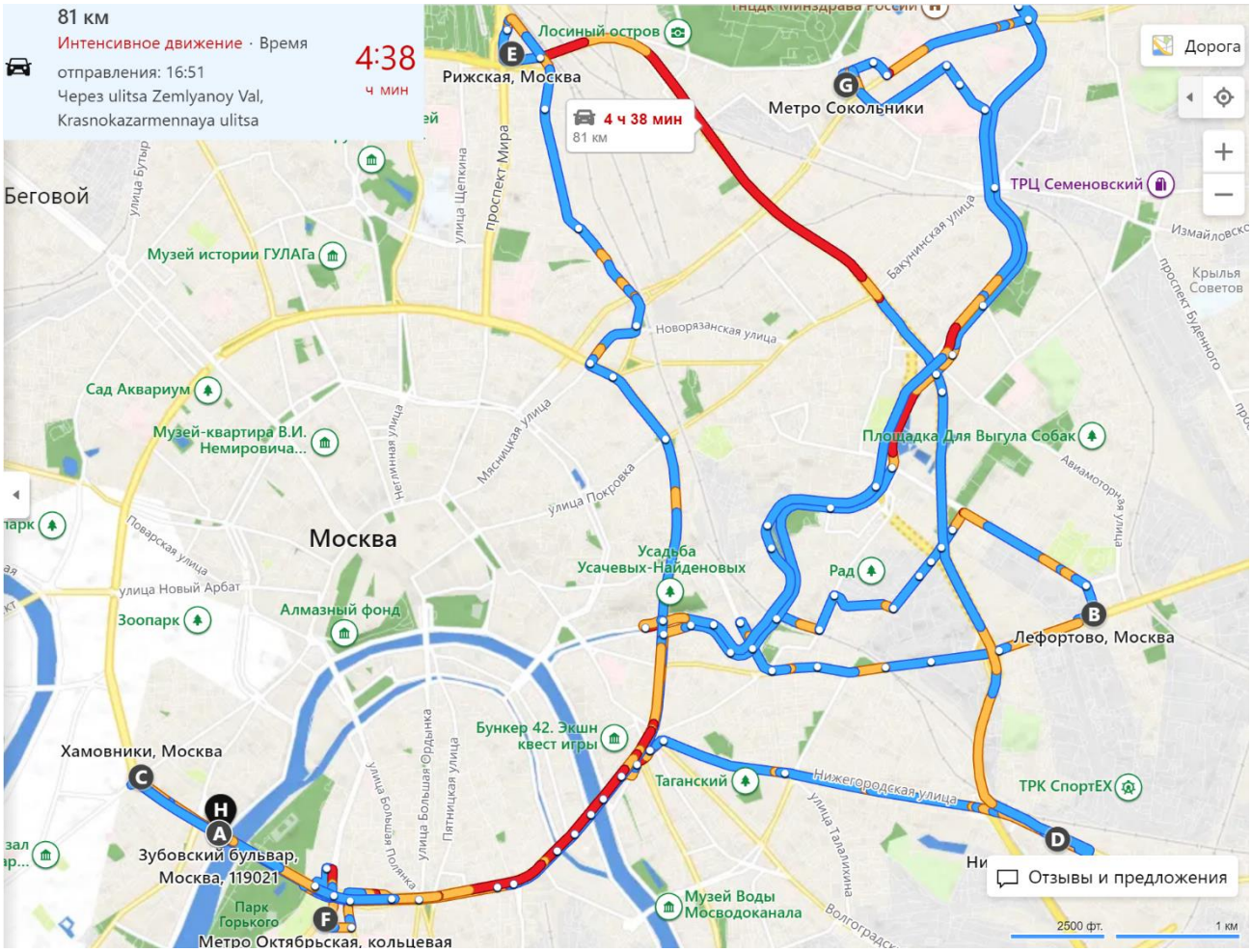

图 1.1 新年参观的乘车路线

图 1.1 中的蓝-黄-红曲线显示了出租车计划在 4 h 38 min 内以 17.5 km/h 的平均速度行驶 81 km 的距离（见图 1.1 的左上角）。 如果您决定仅步行绕城而行（见图 1.2 中的圆圈），则距离会减少到 62 km，但以 5 km/h 的速度需要 12 h 19 min。

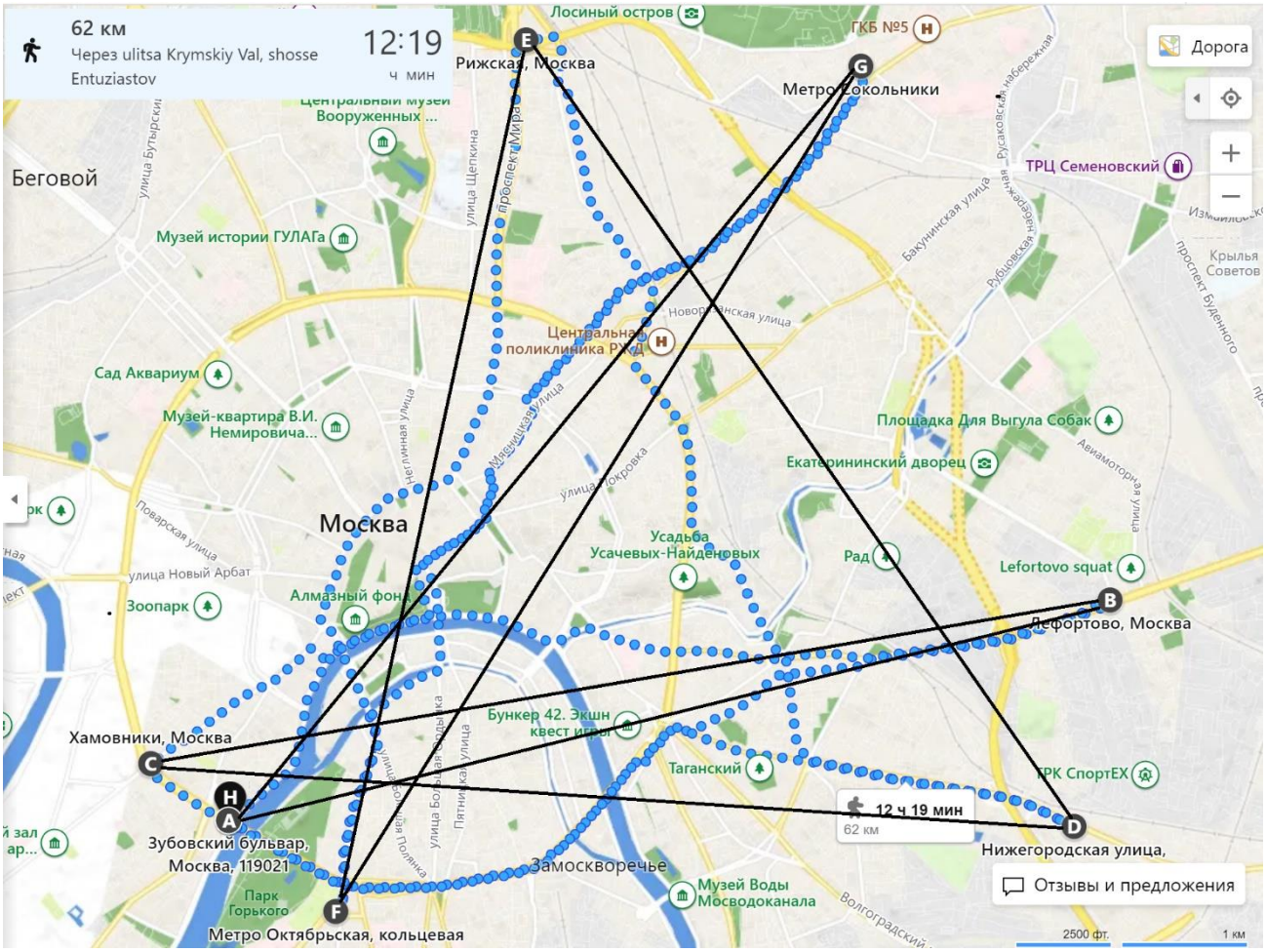

图 1.2 新年参观的步行和直升机路线

很长一段时间以来，人们一直在谈论空中出租车将出现在莫斯科，它可以避免交通拥堵（见图 1 中线条的红色部分）并且可以直线飞行（见图 1 中的黑色直线）图 2)。 毫无疑问，如果契诃夫故事中的主人公有生之年，他会使用这种交通方式。 这是另一个不太奇妙的场景。 假设故事的主人公将他的名片贴在四轴飞行器上，它也直线飞向朋友和熟人，避开交通拥堵，向他们投下新年贺卡或小礼物。 在旧俄罗斯，每逢节假日，官员们都会到老板家，在门卫的签名上签名[在契诃夫的短篇小说《秘密》中，门卫单上的无名小卒的签名产生了非常有趣和滑稽的后果。 读者，也读读这个故事吧!】 特刊。 拜访同龄人但没有找到房子主人的人，将名片留在走廊上。 顺便说一下，这就是这些小白纸板矩形名称的来源。 以下是您可以从果戈理的《死魂灵》中读到的话：“一张名片，即使是写在梅花 2 或方片 A 上，也是非常重要的。 正因为如此，两位女士，好友甚至亲戚，吵架了，正是因为其中一位回访时不知何故吝啬。 无论丈夫和亲戚后来多么努力地试图和解，事实证明，虽然世界上任何事情都可以做到，但只有一件事是不可能的：让两个因为一次拜访而吵架的女士和好。”[俄罗斯文学的经典可以用契诃夫的歌舞杂耍“婚礼”[8]中的另一句话来评论：“他们想展示他们的教育，总是谈论难以理解的事情。” 但作者只是单纯关心工程教育的

人性化……嗯，当然，我也想表现一下自己的博学。 不仅在 SMath 和 Mathcad 中，而且在文献中。]

很明显，出租车路线（图 1.1）和步行路线（图 1.2）远非最佳。 当然，我们可以假设访问的时间是事先约定好的，这可以解释从城市的一端到另一端的如此长的旅程。 但最有可能是这样的：一位新年宿醉的丈夫（见契诃夫的故事）从他的妻子那里收到了一份需要探望的人的名单。 丈夫没有多想，只是茫然地盯着纸条，对出租车司机下达了必要的命令。 如果这是一个巨大的超市，那么其中的路径优化问题也会导致旅行商问题，更准确地说，是流动买家问题，而不是流动卖家问题。] 顺便说一下，这个故事的结尾描述了妻子指责丈夫的另一起丑闻，原因是出租车司机的超支（5 卢布 80 戈比）。

这就是 L.N. 小说中的主人公。 托尔斯泰的《复活》[3]是这样写的：“聂赫柳多夫想好了先去哪里，后去哪里，以免回头，首先去了元老院。” 涅赫柳多夫在圣 圣彼得堡并没有进行空洞的访问——他访问了国家机构的门槛，希望能改变卡秋莎·玛斯洛娃的判决——她是涅赫柳多夫本人的欲望和卑鄙的牺牲品，也是误判的受害者。 与契诃夫故事中的主人公不同，涅赫留多夫在“旅行”之前试图以某种粗略的方式解决旅行商的问题。 顺便说一句，在我们这个时代，骑自行车的送货员解决了这个问题，他们肩上扛着黄色方形袋子里的食物。 时间就是金钱！

让我们编写一个小程序来解决契诃夫故事中的旅行商问题（TSP——来自英语旅行商问题）。 为此要使用什么计算机工具！？ 作者习惯于使用 Mathcad [4]，但是……

在写这篇文章时，俄罗斯发生了一些事情，这种事情在每个世纪初以某种致命的频率发生在它身上。 让我们不要深入研究历史，但是.....

17 世纪初——动荡时期和波兰干预，18 世纪初——北方战争及其重要的波尔塔瓦战役，19 世纪初世纪——拿破仑入侵，20 世纪初——第一次世界大战和内战，21 世纪初是乌克兰危机，何时结束，如何结束还不得而知。 但即使在世纪中叶，也并非一切都是平静的：19 世纪——克里米亚战争，20 世纪——伟大的卫国战争……历史的钟摆朝一个方向或另一个方向摆动（从战争到和平）大约有半个世纪的时间。 我们可以从我们提到的列夫托尔斯泰的作品中研究与拿破仑的战争和克里米亚战争。 21 世纪初俄罗斯与西方的关系史正在等待托尔斯泰的到来。 您喜欢未来小说的标题：“混合战争与数字世界”吗？

最近发生的事件，尤其是西方施加的制裁，使得在俄罗斯无法从 ptc.com 下载完整版 Mathcad 并使用它一个月。 试用月结束后，程序将被截断 (Mathcad Express)，除非购买了完整版的许可证。 但是现在，由于制裁，我们无法下载任何东西，也无法为某些东西付费。 俄罗斯正患上恐俄症[恐俄症是欧洲的一种慢性病，有衰退期和恶化期。 诚然，这种疾病在西方有不同的称呼，他们认为受害的不是西方，而是俄罗斯。 但事实一如既往地位于中间。 在托尔斯泰和契诃夫的作品中，可以追溯到对这种现象的平衡评估，这种评估延续了几个世纪。]。 是的，所有基础知识的基础——Windows 操作系统——对我们来说也悬而未决。

但是，幸运的是，美国程序 Mathcad 有一个名为 SMath (www.smath.com) 的俄罗斯类似程序，它可以成功地应用于解决许多数学和工程问题，而无需使用国外进口程序。 另一个优点是 SMath 程序不仅可以在 Windows（如 Mathcad）下运行，还可以在其他免费操作系统下运行。

那么，让我们下载 SMath，将其安装在计算机上（这在几 min 内完成）并以最简单的方式解决契诃夫的新年访客问题，即最近邻法，其算法属于贪婪算法组 [5]. 在此过程中，我们注意到 SMath 和 Mathcad 之间的区别。 或者更确切地说，不是顺便，而是主要！

图 1.1 或图 1.2 中莫斯科市中心的方案[在契诃夫时代，它几乎是整个莫斯科。]被上传到 Paint 编辑器，以便首先绘制带有箭头的直线，标记空中出租车的路线或 莫斯科上空的四轴飞行器 [我们顺便注意到图 1.2 中沿着某些直线的飞行经过了克里姆林宫。 出于显而易见的原因，这是不可接受的。 因此，这些线必须稍微弯曲。]，其次，将地图数字化——获取计划访问点的坐标，以像素为单位。 为此，将光标依次移至每个点，其坐标写在 Paint 的左下角。 这些坐标被转移到 X 和 Y 向量（见图 1.3）。 从 Y 向量的元素值中减去一个常数 - 数字 1456。这是我们放置在 Paint 环境中以进行数字化的两张图片的高度（以像素为单位）。 事实是，在 Paint 中，坐标原点是在图片的左上角，而在我们以后的图表中是在左下角。

图 1.3 显示了计算的开始——输入访问者路线开始的点数（稍后我们将对任务进行一些概括，并假设旅行可以从任何点开始，并且可以有任意数量的点） . 变量的起始索引 i 是通过点 i.start 输入的——就像在良好的旧版本中一样[许多用户出于多种原因不切换到 Mathcad 的新版本，即 Mathcad Prime，由于 到其不寻常的界面和缺少旧版本的某些功能。] Mathcad（15 及以下）。 变量的字体自动从直接变为斜体，强调这不是内置的，而是用户定义的对象。 在 Mathcad Prime 中，当输入文本索引（变量名称的部分向下移动）时，分隔点被放弃。 在那里，通过按下标有 a2 的特殊按钮输入索引。 再次按下此按钮将停止索引输入。 这使您可以在计算中使用诸如 H2O、H2SO4 等奇异变量。在 SMath 中有这样的机会是可取的。

我们立即注意到，在 SMath 环境中，没有 Mathcad 的 ORIGIN 系统变量，它设置向量中元素的初始数量、矩阵中的行和列，默认情况下（出厂设置）为零。 在 SMath 环境中，数组仅从一开始编号。 这是好是坏是另一回事。 在我们的计算中，istart 变量被分配了一个——游客从第一个点开始他的旅行——从祖博夫斯基大道上的房子。 然后我们将 istart 的值从一个值更改为另一个值，看看会发生什么。

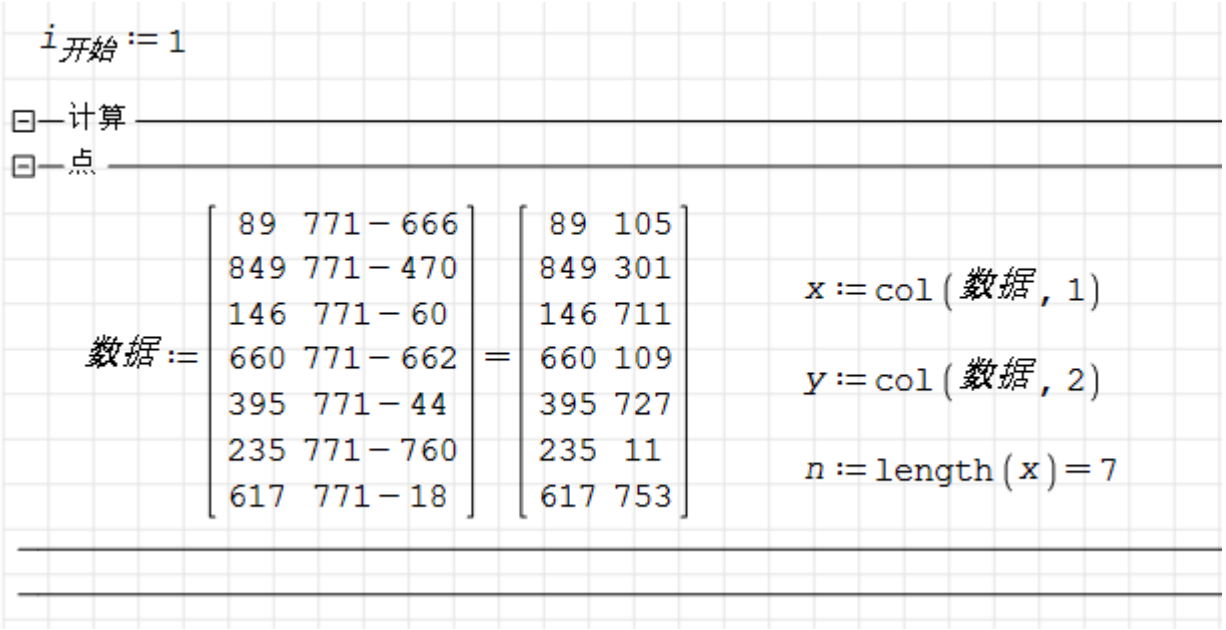

图 1.3 开始计算旅行商的路径——契诃夫的访客

因此，在计算的最开始，我们按小写的 i 键，得到如图 1.4 所示的结果。 SMath 包在下拉列表中列出了名称以此字母开头的所有内置和用户定义的结构。 这个很方便，Mathcad 没有。 如果需要，可以通过菜单命令（开关）“查看/动态输入辅助”禁用此选项。

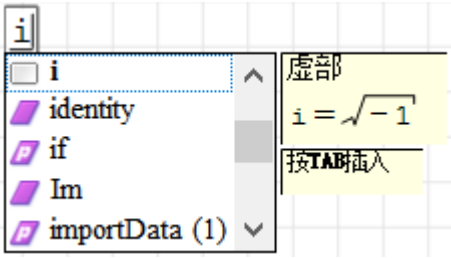

图 1.4 名称以字母 i 开头的变量和函数列表的开头

创建文档时，建议立即使用所谓的**折叠条（area）**，其中可以放置一部分计算。 这样的区域可以折叠，这样您就可以同时看到文档中彼此相隔很远的部分——例如，输入初始数据和答案。 如果没有区域，您将不得不为此转动鼠标滚轮，这对于大量计算来说是不方便的。 因此，单击带负号的正方形（见图 1.3），该区域会折叠。 减号变成加号，您需要的东西（初始数据和答案）就出现在您的眼前。 再次单击加号，该区域将打开，显示其中的内容。 可以为区域命名，如有必要，可以用密码保护——防止窥探[在 Mathcad 和 SMath 用户论坛上，忘记密码的人经常询问如何打开密码保护区域。]。 在 SMath 中，您可以在一个区域中插入另一个嵌套区域，这样您就可以构建计算。 这在 Mathcad 中不可用。 下面，图 1.4 显示了一个名为 **计算** 的“主要”区域，其中嵌套了四个命名区域，其中存储了用户定义的函数，这将在后面讨论。

作者对他的画作了一点梦想。 他不仅在区域的开始处，而且在区域的末端用方块中的指针绘制正负值——这是在 Mathcad 15 中完成的方式，但在 Mathcad Prime 或 SMath 中

都没有。 另外，为了固定计算结构，作者将嵌套区域向右移动。 Maple 包中提供了类似的东西。 作者将这个幻想（关于一个愿望）告诉了开发者 SMath，SMath 承诺在下一个版本的包中实现这样一个方便的功能。

在图 1.3 中名为 Points 的嵌套区域的前两个语句中，将具有七个元素的向量分配给变量 X 和 Y。这与 Mathcad 15 中的方法相同——按下带有正方形图像的按钮 Matrix 面板上的二乘二矩阵，如下图 1.5 所示（该面板位于 SMath 工作窗口的右上角）。 之后，在出现的对话框中，请求矩阵的维度——行数和列数。 但是，遗憾的是，在此窗口中没有用于删除行和列或添加它们的按钮。 当然，这个缺陷需要改正，更好的是像 Mathcad Prime 中提供的那样，可以通过拖动鼠标来选择未来矩阵或向量的大小（大约是这样做的 在 Word 和 Excel 中），还有用于删除不需要的元素和插入其他元素的按钮。 必须以相当棘手的方式删除 SMath 矩阵的行和列。

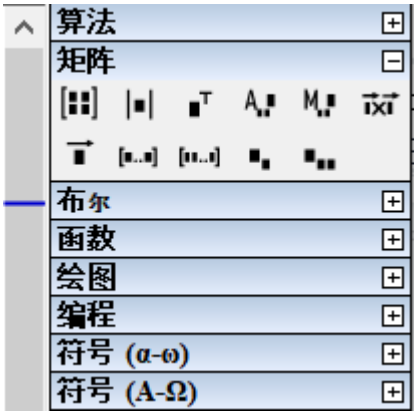

图 1.5 矩阵操作面板

请注意，SMath 包还允许您从磁盘上的文件下载数据。 通常，有些程序会在您用鼠标单击图片时将点的坐标作为文本文件写入磁盘。 这样的一系列数字对可以自动传输到 Smath，以便进一步处理数据。

图 1.6 显示了一个带有用户函数的扩展区域，该函数返回一个方阵，其中包含旅行商路径上各点之间的距离。 曾几何时，在乡村公共汽车上可以看到类似的矩阵（tablet）。 上车的乘客告诉售票员他要去哪里。 售票员看了看矩阵，将车费通知了乘客。 这种总线矩阵的对角线上有破折号。 在 SMath 中我们把无穷大的值存在那里，为了防止售票员直接从一个点到自己！

在图 1.6 的程序中，可以看到三种编程结构：程序块、局部变量和程序控制结构。 SMath 环境中的程序块（Pascal 语言中的开始-结束结构）由使用行按钮输入的垂直条固定（图 1.7）。 如果变量被输入到程序主体中——在垂直条的右侧，变量将自动成为局部变量（仅在程序中可见）。 我们有三个这样的变量 i、j 和 M。请注意，在 SMath 中，冒号和等号运算符 (:=) 用于输入全局变量和局部变量。 你只需要按“冒号”键，“等号”就会自动加上。 更好的是，按等号键而不是冒号键。 如果变量是自由的，那么“等号”将变成“冒号和等号”，用户将有机会在变量中输入所需的值。 如果变量已经存储了一些东西，那么按下“等于”键将导致显示这个东西。 Mathcad 使用不同的运算符进行局部赋值，即

左箭头，这很不方便并且没有任何逻辑。 完整的距离矩阵（四舍五入为整数）的内容显示在图 6.1 的顶部。

Function M

$$M(X, Y) := \left| \begin{array}{l} \text{for } i \in [1..n] \\ \quad \text{for } j := 1, j \le n, j := j + 1 \\ \quad\quad \text{if } i \ne j \\ \quad\quad\quad M_{i\,j} := \sqrt{\left(X_i - X_j\right)^2 + \left(Y_i - Y_j\right)^2} \\ \quad\quad \text{else} \\ \quad\quad\quad M_{i\,j} := \infty \\ M \end{array} \right.$$

Function M

图 4. 返回距离方阵的名为 M 的函数

在 SMath 环境中改变运算符的自然执行顺序（从左到右，从上到下）的程序控制结构集与 Mathcad 中的大致相同（图 1.7）。 仅缺少 return 语句。 SMath 环境中的 on error 运算符（错误处理）的名称为 try（尝试）。 相同的重命名适用于 Mathcad Prime 环境

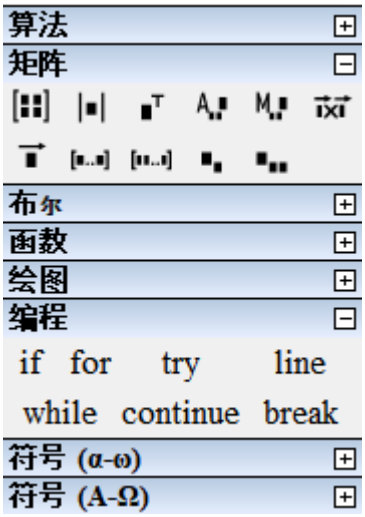

图 1.7 操作面板编程

带有 for 参数的循环在 SMath 中以两个版本实现——具有三个和四个操作数——见图 1.8

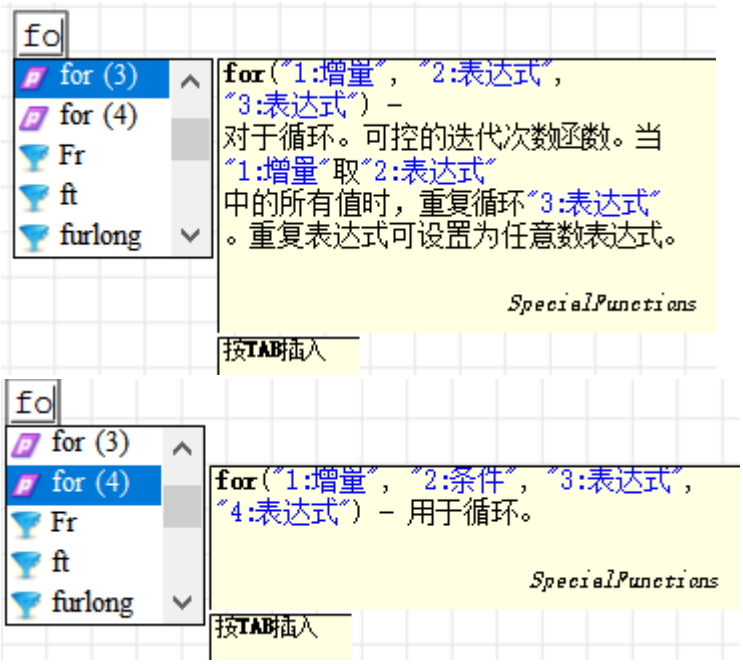

图 1.8 for 语句的两种形式

在图 1.8 中的程序中，使用了 for 循环的两种变体，尽管我们可以将我们限制为仅使用具有三个参数的第一个版本。 但在某些情况下，第二种选择更可取。 例如，当用 C 编写的程序被翻译成 SMath 时，就会发生这种情况[我们邀请读者在 Internet 上找到并在 SMath 中实现一个 C 程序，该程序使用更先进的方法——退火法、蚁群法、遗传学法——解决旅行商问题 等]，其中使用了 for 循环的四参数形式。 此外，这种形式使得在 for 循环体中改变循环参数的值是合法的。

第一个（外部）for 循环包含循环参数的名称和“属于”符号。 然后通常在这里插入所谓的范围变量（Range），使用等差级数。 这个级数的空白在 **Matrix** 面板（图 1.5）中有两个版本——带有范围变量的第二个值 [1.1, 1.2 .. n] 和没有它 [1 .. n]，如图 1.6 所示。在 第二种情况，当没有指定步长时，范围变量的第二个值比第一个大一个（步长将为一个）。 或者如果范围变量的第一个值小于最后一个值，则小于第一个（减一）。

我们立即强调 SMath 环境中的范围变量是一个成熟的向量，所有向量运算都适用于它——参见图 1.9 中的示例。 不幸的是，这在 Mathcad 中不可用。 在那里，如有必要，您将不得不执行特殊的复杂过程以将 Range 变量转换为完整的向量。

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

图 1.9 将范围变量用作完整向量的示例

这是 SMath 的另一个有用属性，坦率地说，应该在一开始就提到它。 从图 1.9 可以看出，十进制数使用的是逗号，而不是点，函数中的参数分隔符不是逗号，而是分号。可以通过调用工具菜单中的选项对话框来选择计算中的点、逗号和其他“标点符号”，如图 1.10 所示。 在这里，俄语版的 Excel 立即浮现在脑海中，数字中有逗号，列表中有分号。 在准备文章时，程序编号中的逗号很方便，因为俄罗斯编辑需要使用逗号，而不是非整数中的点。

图 1.10 菜单选项

图 1.11 显示了一个用户函数，它返回向量的最小元素的索引。 使用了矢量元素的简单枚举。 在 Mathcad 中，在数组（向量和矩阵）中搜索所需的索引是通过具有两个参数的内置匹配函数执行的，该函数返回所需的索引值数组。 图 1.11 中的函数返回一个标量，即使参数向量包含多个最小值也是如此。 这对我们来说已经足够了，但是 match 函数和其他类似函数在 SMath 环境中也会很有用。

在图 1.11 中，该函数在创建函数后立即被调用以检查其性能。 可以看出，在向量 X（契诃夫来访者的旅行点的横坐标——见图 1.1-1.3）中，最小元素 212（这里涉及到内置函数 min）排在第三位。 如果有多个这样的元素，那么内置的 Mathcad 匹配函数将返回一个向量。 在我们的例子中，这种情况下的 i_{min} 函数将返回一个标量——最小值系列中的最后一个值。

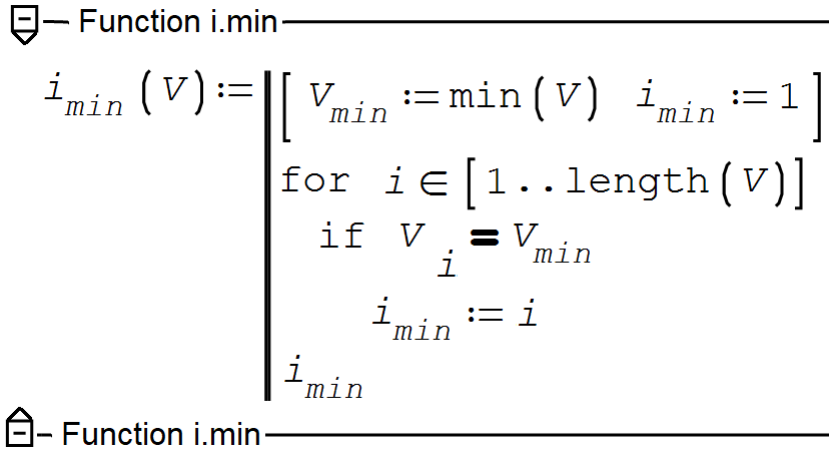

图 1.11 返回向量最小元素索引的函数

因此，已经创建了辅助函数，并且可以将主函数引入计算中——该函数返回一个向量，该向量包含销售员必须经过的点数，由最近邻算法指导——见图 1.12。它是 很简单，不需要太多解释。 它有两个 for 循环，第二个循环嵌套在第一个循环中（另请参见图 1.6）。在嵌套循环中，形成一个向量 S，它存储从当前第 j 个点到其余路线点的距离。 使用 i_{min} 函数找到的该向量的最小元素的索引成为执行语句 $Way_i := i_{min}(S)$ 后旅行商路径上的下一个点的编号。 之后，从第 i 点到第 j 点的距离矩阵的相应元素被赋予无穷大的值，使得销售员不再沿着这条路线走一遍。 在图 1.13 中，您可以看到执行最近邻算法前后距离矩阵 M 中包含的内容。

Function Way

$$Way(X, Y, i_{start}) := \left| \begin{array}{l} \left[M := M(X, Y) \qquad Way_1 := i_{start} \right] \\ \text{for } i \in [2..n] \\ \quad \left| \begin{array}{l} \text{for } j \in [1..n] \\ \quad S_j := M_{Way_{i-1}, j} \\ Way_i := i_{min}(S) \\ \text{for } j \in [1..(i-1)] \\ \quad M_{Way_i, Way_j} := \infty \end{array} \right. \\ Way \end{array} \right.$$

$$Way(X, Y, 1) = \begin{bmatrix} 1 \\ 3 \\ 6 \\ 4 \\ 2 \\ 7 \\ 5 \end{bmatrix} \qquad Way(X, Y, 2) = \begin{bmatrix} 2 \\ 4 \\ 6 \\ 1 \\ 3 \\ 5 \\ 7 \end{bmatrix}$$

Function Way

图 1.12 用最近邻法解决旅行商问题的程序

图 1.12 还显示了为两个不同的起点调用 Way 函数的示例（另请参见图 1.15）。 在图 1.12 中的程序中，如果程序末尾的单词 Way 被字母 M 替换，则返回的矩阵将沿主对角线返回原始“无穷大”，并在后面添加“无穷大” 最近邻方法的实现。

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

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

图 1.13 最近邻算法执行前（上）和执行后（下）距离矩阵的内容

在图 1.5 中，在 **Matrix** 面板上，可以看到两个按钮，用于访问向量和矩阵的元素。 在 Mathcad 环境中，一个按钮就是为此而设计的，矩阵索引彼此之间不是用空格分隔，而是用逗号分隔。 图 1.12 中的函数在使用索引（文本（imin、istart,）和数字）以及返回向量或矩阵元素的运算符方面堪称杰作。 我们的一些运算符是“三级”的，例如，矩阵 M 的第一个索引是 Way 向量的索引。

图 1.14 显示了包含四个嵌套折叠区域存储点数据和上述三个用户定义函数的完整计算。 它仍然只是构建一个图形并对其进行动画处理。 为此，我们使用图 1.14 底部所示的运算符。请注意，绘制图形（2D 和 3D）的技术在 SMath 中与 Mathcad 中的技术根本不同。

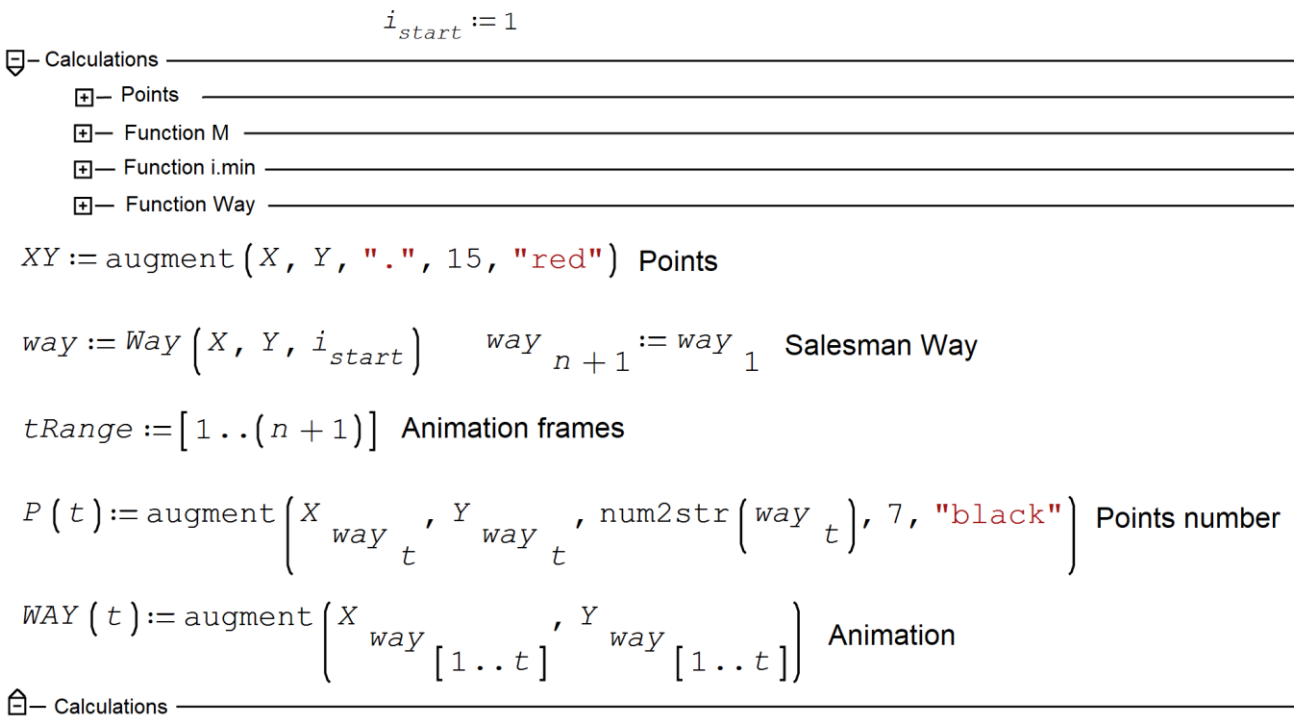

图 1.14 操作员需要创建旅行推销员的图形和动画

在图 1.14 中，XY 矩阵由 n 行和以下五列组成：向量 X、向量 Y、点（“.”），大小为 15 个单位，颜色为红色。 除了点之外，您还可以使用叉号（字母 x 或“+”）、圆圈（字母 o）和星号（“*”）。 其他标点符号和字母（俄语和拉丁语）写在图表上指定位置的右侧和下方。 这个很方便，Mathcad 没有。我们在给旅行商经过的点赋编号的时候会用到这个（图 1.14 倒数第二个算子）。

在 Mathcad 中，动画由 FRAME 系统变量控制。 在 SMath 环境中，这项工作由变量 t 完成，它必须作为动画对象的参数。 我们有 P(t) 和 WAY(t)。 动画帧数必须存储在向量或范围变量中。 对我们来说，这将是 tRange 变量——参见图 1.17 中的对话框。 可以通过在激活的图表上单击鼠标右键来访问此窗口。 通过图 1.16 所示的菜单命令将图形插入到工作文档中。 动画帧率等动画参数通过图 1.18 所示的对话框设置。

图 1.15 显示了来自两个不同起点的契诃夫访客的路径——从第一个（Zubovsky 大道上的访客的房子）和第二个（Lefortovo——另请参见图 1.12）。 在动画期间（可以在此处 https://community.ptc.com/t5/Mathcad/Mathcad-vs-SMath/td-p/802499 的第 6 点中看到）一条蓝色直线从起点（第一个 或第二个）并“绕过”其他红点，在左侧和下方显示其编号。

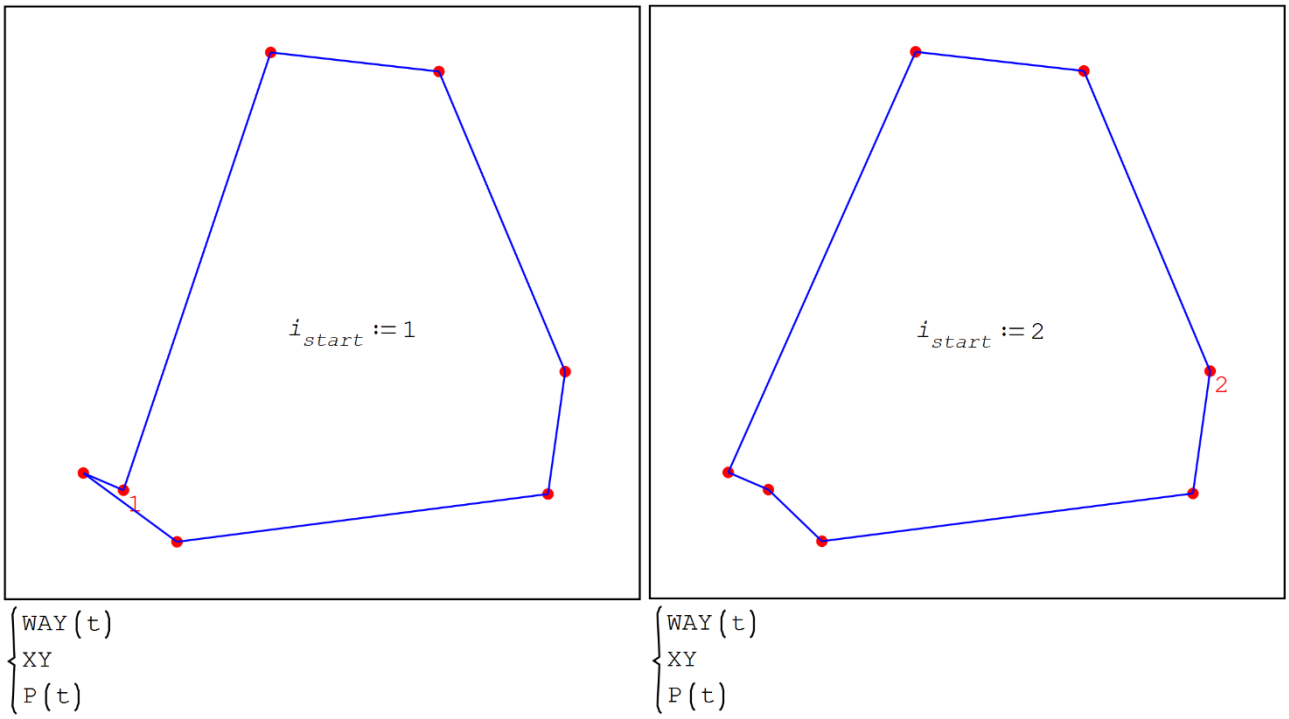

图 1.15 契诃夫故事中来访者从两个不同起点的路径。

为了尽量减少路线的长度，我们的契诃夫访问者不得不从 Zubovsky 大道（第 1 点）而不是 Lefortovo（第 2 点），而是去 Khamovniki（3），然后到卡卢加门（6），然后到 Nizhegorodsky 火车站 (4)，然后是 Lefortovo (2)。 此外，他的路线必须经过 Sokolniki (7)、Krestovskaya Zastava (5) 并在 Zubovsky Boulevard (1) 回家。 但图 8 中的右图显示了一条更短的路线：在 Khamovniki，必须不要贪婪，不要去最近的 Kaluga Gates，而是去更远的 Krestovskaya Zastava。 原则上，旅行商问题的最优解应该独立于起点。 在我们的例子中，情况并非如此——见图 1.15。

但所有这些都适用于由直线段组成的路线（直升机旅行——见图 1.2）。 实际上，计算机或智能手机上的应用程序必须为订购出租车的人提供环形路线（见图 1.1），以通过解决旅行商问题来优化行程，这不仅要考虑停靠站的位置， 还有交通状况，不仅减少了行驶距离，还减少了旅行成本。

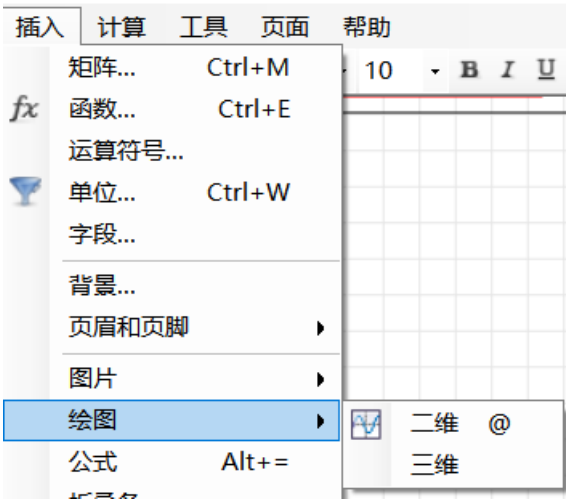

图 1.16 用于插入二维绘图的菜单命令

图 1.17 设置动画变量

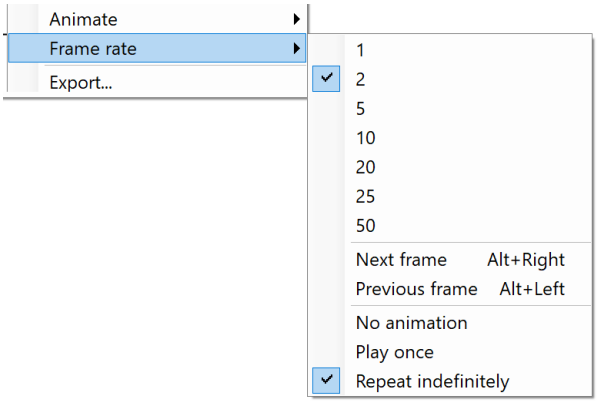

图 1.18 动画选项

图 1.19 显示了一个旅行售票员移动超过 50 个随机点的三个动画帧，这些随机点可以通过返回随机数的函数输入到 X 和 Y 向量中。

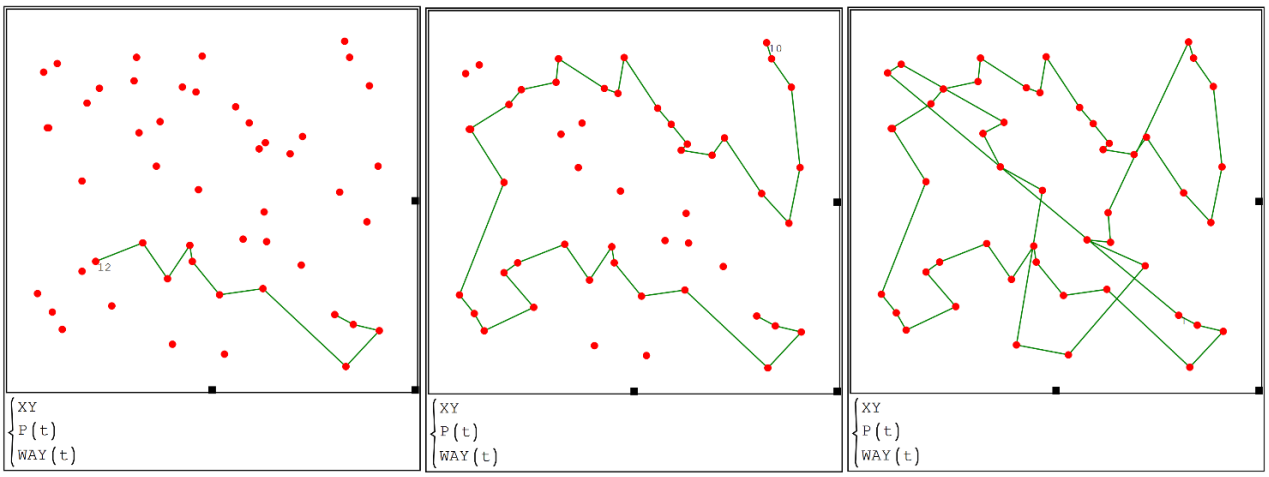

图 1.19 旅行商沿五十个随机点移动的三帧动画

如果图 1.12 中的程序要用 max 函数替换 min 函数，并将无穷大符号替换为负无穷大，那么将实现最远邻居算法 — 请参见图 1.20。您可以遍历所有起点并选择一个 比契诃夫故事里妻子送丈夫去的那条路还要远！

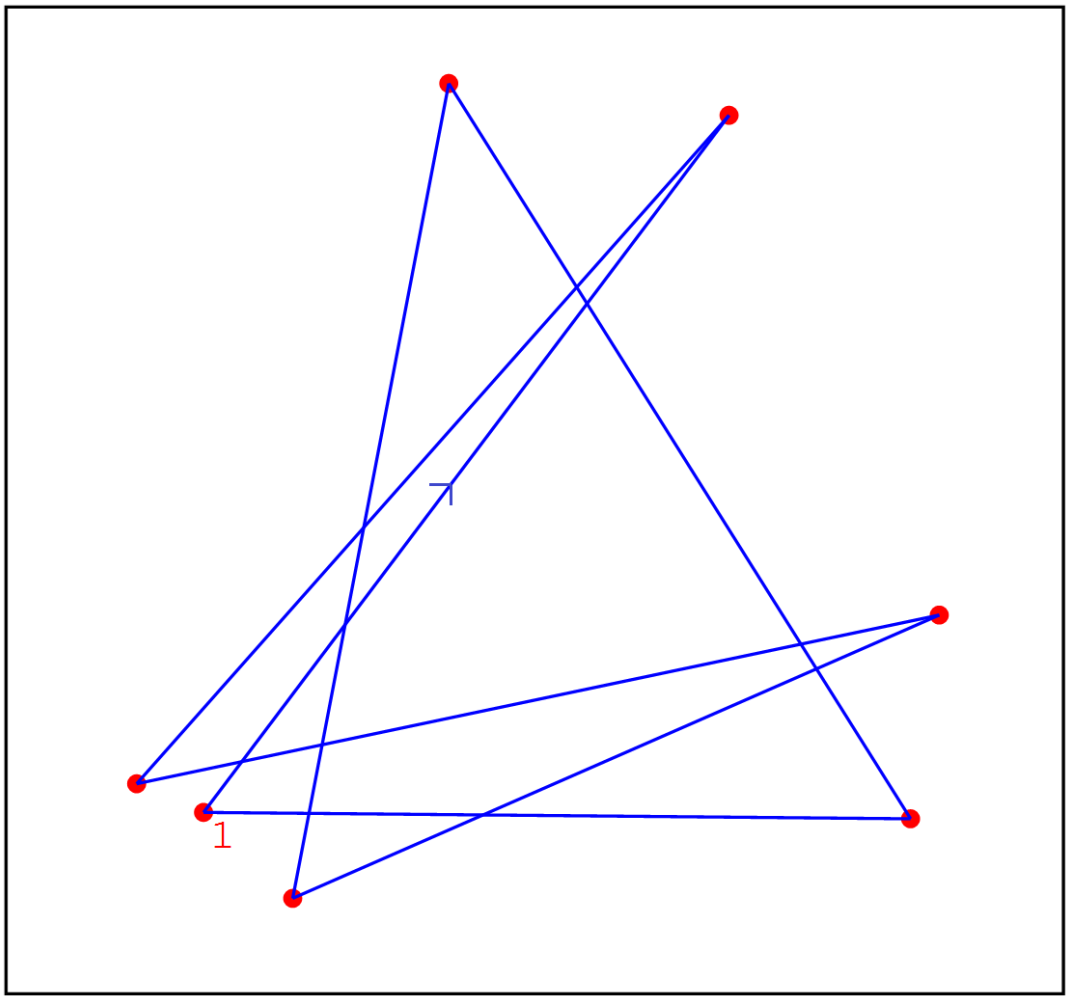

图 1.20 根据最远邻居算法的访客路径

图 1.21 显示了维基百科文章“旅行商问题”中的一张图片，其中解释了尝试通过蛮力解决旅行商问题的结果。 Frau 派她的 Herr 和朋友们坐飞机飞遍德国的 15 个城市.....

图 1.21 解决德国 15 个城市的旅行商问题。

站点［7］包含有关旅行商问题的大量信息。 特别是，在那里你可以找到一个旅行商人走过意大利（图 1.22）和其他国家乃至整个世界的所有定居点的路径。 当然，使用了更高级的算法，消除了环路，真正做到了路径最小化。 但是这些路线还没有被证明是最短的。 我们建议读者在 SMath 中实现此类算法——创建一个新的 Way 函数。 所有这些算法都是启发式的，使用不保证准确或最佳但足以解决问题的实用方法。 但是对于少量的点（图 1.15），即使不使用复杂的算法也可以找到绝对准确的解。

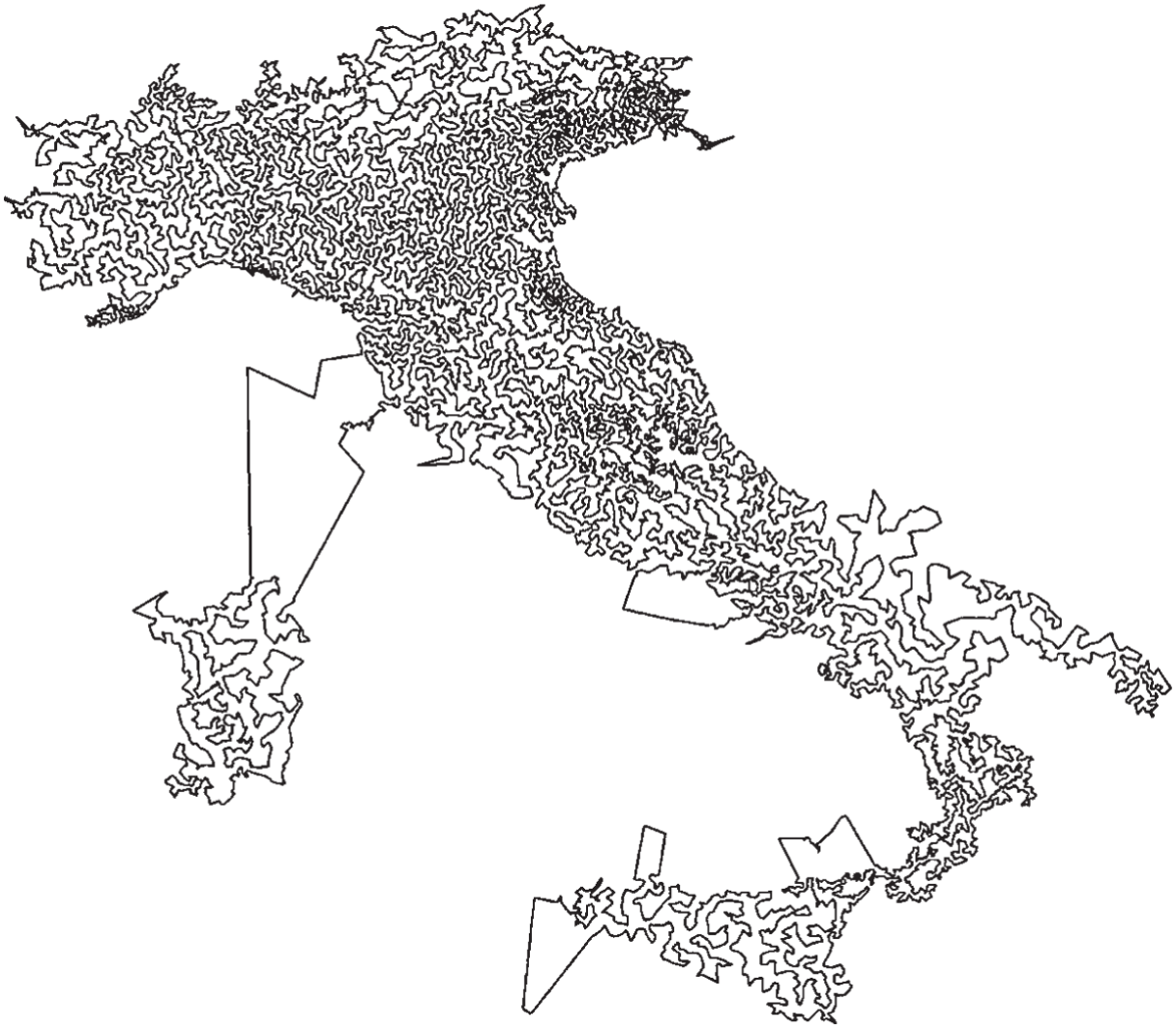

图 1.22 推销员穿越意大利

章后记

2023 年 2 月，当本书的工作即将完成时，本教程的第一作者收到了来自加拿大的 Maple 数学包的开发者——来自 MapleSoft 的消息，这是一个数学程序的测试版，其中包含解决问题的工具 旅行商问题出现了。 作者立即想在 Chekhov 的访客问题上测试这些工具——见图 1.23-1.25。 这就是他所做的！

图 1.23 显示以下内容：

• with(GraphTheory) 命令加载用于在 Maple 环境中处理图形的工具。

• 在 Points 变量中，用户编写了一个具有七行四列的矩阵，用于存储有关 Chekhov 访问者旅行的信息。 要获取此数据，应将行程示意图（图 1.1 或 1.2）放置在例如 Paint 图形编辑器环境中，以便可以将光标从一个点移动到另一个点以读取以像素为单位的坐标。 Points 表其实不是一个表，而是一个最简单的相对论数据库，有四个字段（标示城市地点的字母、地名、地名、横坐标和纵坐标）和七个条目。 城市中的地点坐标也可以通过以下方

式获取——在电脑上打开一张城市地图，将鼠标光标放在所需地点，读取其坐标：北纬和东经。 但是，随后需要将这些角度转换为平面地图上的千米数。 不是将点作为数据库输入，而是可以将它们作为具有空主对角线的方阵输入，该对角线存储点之间的距离或它们之间的行程时间（成本）。 如果这样的矩阵 mi，j 的一个元素存储值 100，那么这意味着 100 是第 i 个和第 j 个点之间的距离，或者旅行时间，其成本等。不仅主对角线 在这样的方阵中是空的，但也有一些其他元素，这意味着两个城市之间的直接旅行是不可能的。
CompleteGraph 命令通知用户我们正在处理一个无向图，该图有 7 个顶点（莫斯科的地点）和 21 条边（连接这些地点的路径）。 有向图的一条边只能在一个方向（单向街道）上“行进”。

• 我们图形的顶点和边由 DrawGraph 命令构建。 结果是一个正七边形，其所有顶点排列成一个圆（style=circle）并由直线连接。 在这些边缘线上，您可以选择标记点之间的距离、旅行时间或费用。

```
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]])
```

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

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

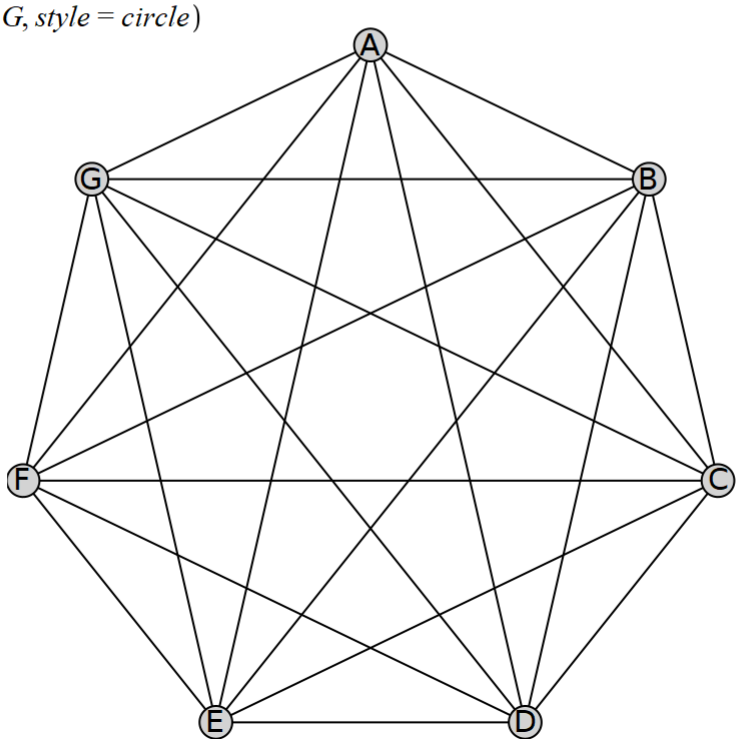

图 1.23 解决 Maple 2023 中的旅行商问题（开始）

TravelingSalesman 函数（图 1.24）首先返回 Chekhov 访问者旅行的距离（6035 像素），其次返回他的路线，沿着这条路线该距离将是最小的。 图 1.24 中的其他两个操作员构建了 Chekhov 的访客的路线，或者更确切地说，是他发射的四轴飞行器。 这些点与图 1.1 和图 1.2 中所示的点相同，只有一个区别——它们在垂直和水平方向上颠倒了。 莫斯科的南边原来是北边，西边是东边。 但这并不重要。 最主要的是问题已经解决：从 Zubovsky 大道（A 点）有必要去（直线飞行）到 Khamovniki（C）或 Kaluga Gates（F），然后“与所有 停止”。 契诃夫的来访者“旅行”到了列福尔托沃（B）。 最小路线将是从任何方向的任何点出发。 但是，如果你只能沿着某个边缘朝一个方向行驶，那么这最后的陈述将证明是错误的。

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

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

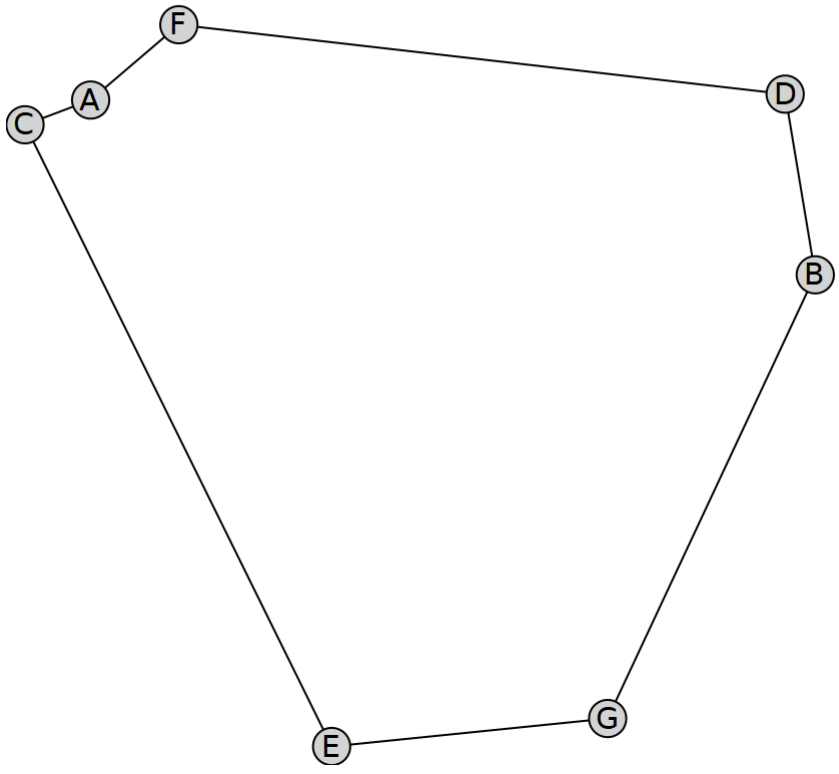

图 1.24 解决 Maple 2023 中的旅行商问题（续）

图 1.25 显示了原始图，在其上用红色粗线额外绘制了具有优化路线的所谓哈密顿环（cycle）（参见 https://en.wikipedia.org/wiki/Hamiltonian_path）。

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

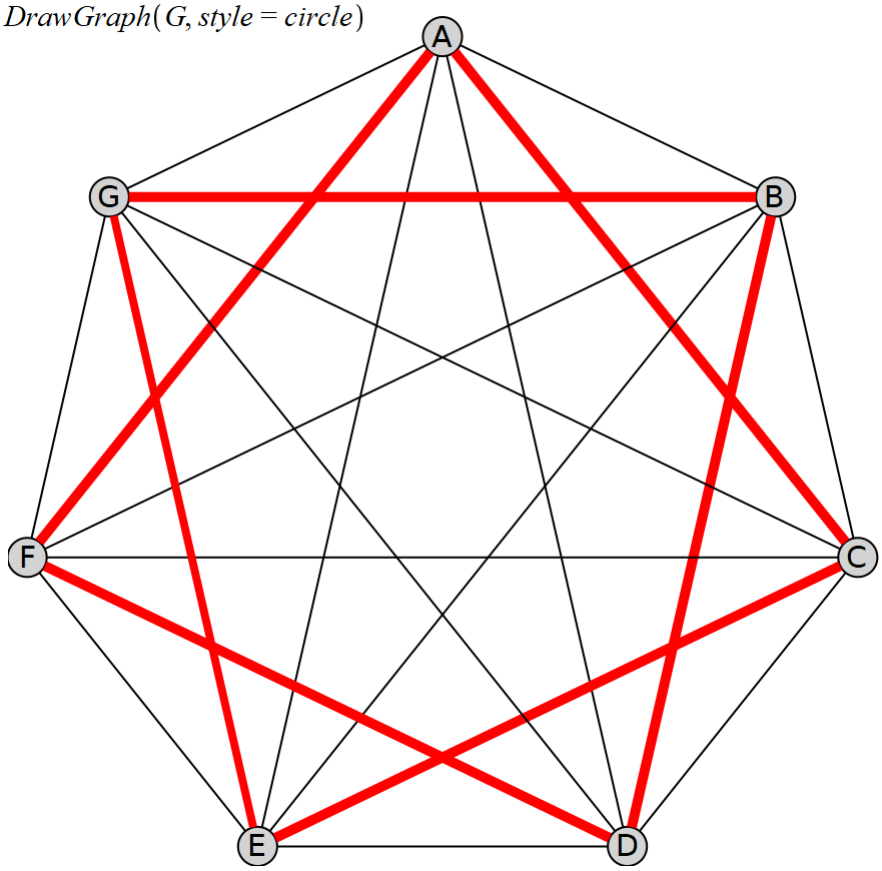

图 1.25 解决 Maple 2023 中的旅行商问题（结束）

使用 Maple 包解决契诃夫故事中的问题，当然是向麻雀开炮（等效的中国短语是：“杀鸡焉用牛刀”）。 只需查看莫斯科地图即可得出最佳路线。 但是随着点数的增加，电脑也不能少了。 例如，旅行推销员路线及其包含 20 个点的图形可能如下所示 - 参见图 1.26

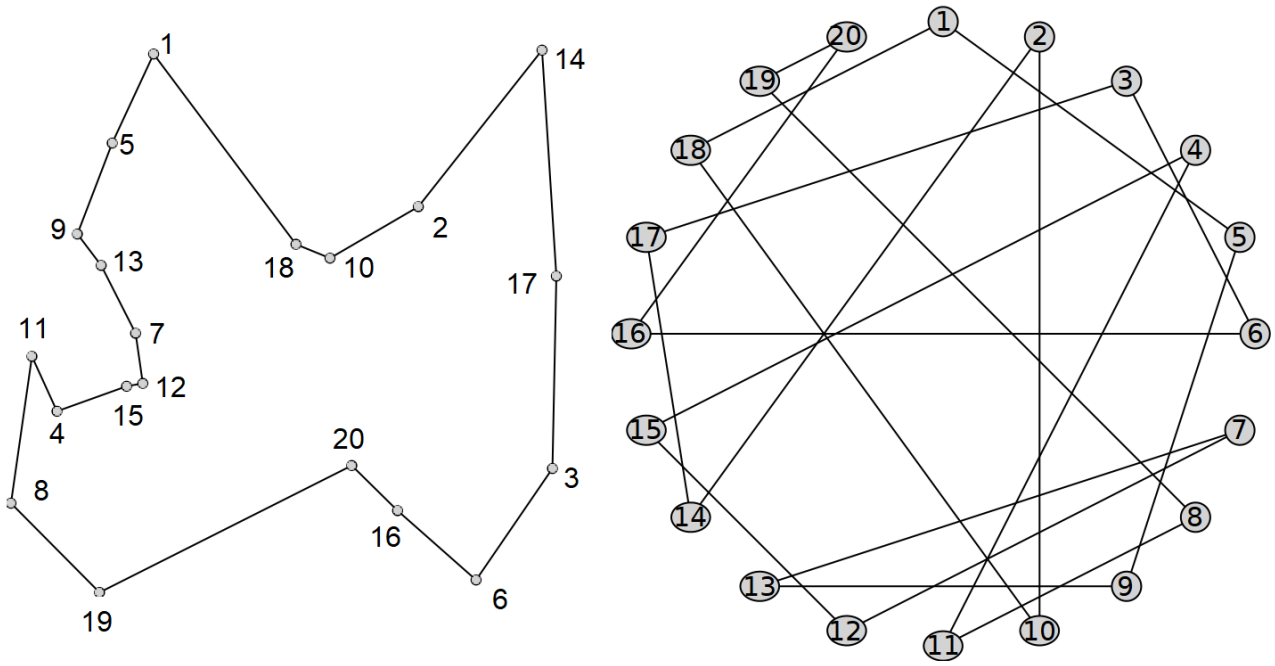

图 1.26 解决 Maple 2023 中的旅行商问题（选项 2）

Maple 包的帮助描述了一个更复杂的任务，如果没有计算机也无法解决这个任务——乘飞机沿着非洲 100 个最大城市的最短路线飞行——见图 1.27

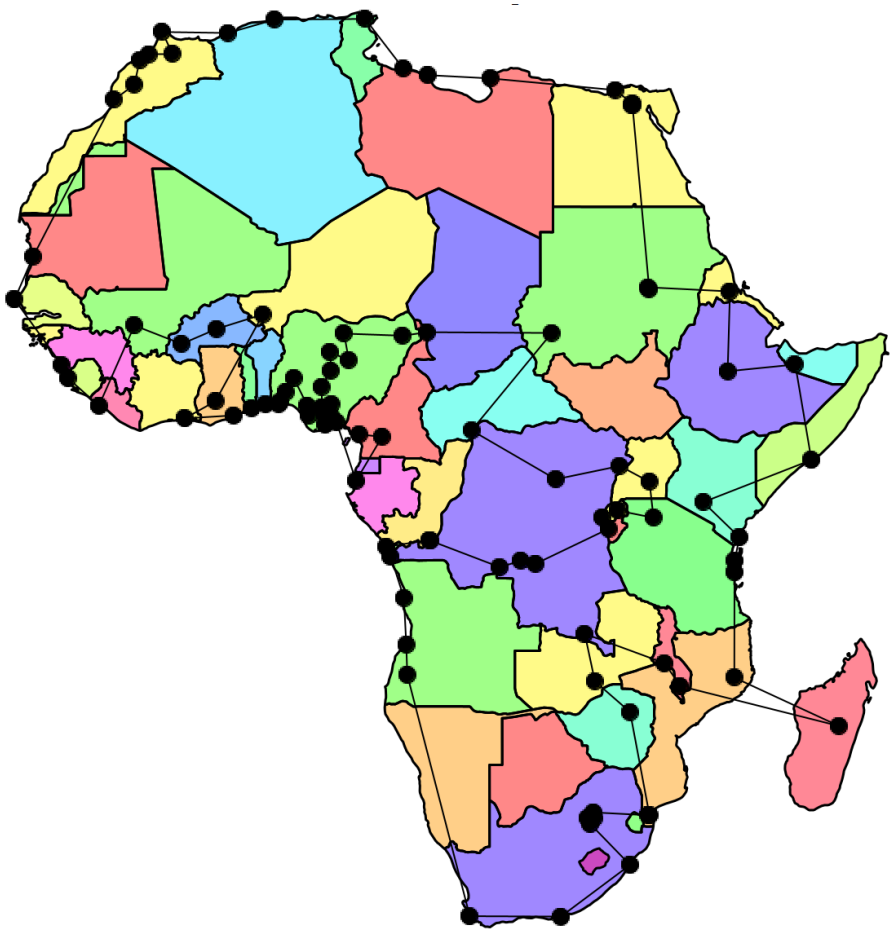

图 1.27 Maple 2023 环境（非洲一百个城市）中旅行商问题解决方案的图形显示

顺便说一下，非洲地图让我们想起了另一个与图论间接相关的有趣优化问题。 这就是可以用来绘制世界政治地图的最少颜色数的问题。 事实证明，只有四种颜色就足够了，如图 1.27 所示 但是在科普杂志中，类似于读者手中拿着的杂志（在他的计算机、平板电脑、智能手机上打开），等高线图经常出现 在愚人节的问题上，他们不能只涂四种颜色的声明.....

让我们回到旅行商问题。

事实证明，应该通过结合交通方式来尽量减少距离、时间或金钱在城市中的移动。 您需要骑自行车，定期将自行车卸下或推入公共交通工具（电车-公共汽车-地铁），甚至乘坐出租车。 我们现在亲眼目睹了这一点，例如在莫斯科，骑自行车的人肩上扛着黄色方包，正以极快的速度比赛。 他们向顾客提供杂货和现成的饭菜，大流行病过后，莫斯科人“沉迷于”这些东西。 在这些商业骑车人的手中，你可以看到一部智能手机，它不仅能指路，还能优化路线，解决了一个相当困难的旅行推销员问题……

另一个例子。 一项世界著名的快递服务，用轮船、飞机、火车、汽车、自行车等方式在世界各地运输货物，聘请数学家和程序员优化运输。 数学家和程序员创建了一个程序，使该服务（公司）每年节省近 20 亿欧元。

而且这样的例子还有很多。

习题：

1. 重新创建 SMath 中文章中描述的计算。

2. 在计算中输入 50、100 或更多随机点，选择初始点并使用最近邻算法从中构建旅行商路线。 然后遍历所有点作为起点并找到最小路线。

3. 通过在 Internet 或其他来源中查找他们的算法，在 SMath 环境中实施更优化的旅行商路线。

参考文献：

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