中国科学: 信息科学 2015 年 第 45 卷 第 8 期: 1060–1079
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE CHINA PRESS
论 文

一类求解 TSP 构建型算法的通用改进策略

饶卫振[①*], 王新华[①], 金淳[②], 刘锋[③]

① 山东科技大学经济管理学院, 青岛 266590
② 大连理工大学管理与经济学部, 大连 116023
③ 东北财经大学管理科学与工程学院, 大连 116025
* 通信作者. E-mail: raoweizhen@163.com

收稿日期: 2014–12–15; 接受日期: 2015–02–07; 网络出版日期: 2015–07–24
国家自然科学基金 (批准号: 71271041)、山东省优秀中青年科学家科研奖励基金 (批准号: BS2014SF001) 和山东科技大学人才引进科研启动基金 (批准号: 2013RCJJ020) 资助项目

摘要 分析了求解旅行商问题 (traveling salesman problem, TSP) 的 4 种贪婪构建型算法的特点, 发现这类算法的最大缺点是在求解初期以贪婪的方式构建解, 导致求解后阶段为初期的贪婪付出代价. 本文在 Held Karp 模型基础上提出了一种改造 TSP 问题距离矩阵的方法 —— 距离矩阵方差最小法 (minimizing variance of distance matrix, MVODM), 以降低这类算法构建初期的贪婪性, 提高算法的总体求解质量. 分别采用 4 种贪婪构建型算法结合和不结合 MVODM 两种方式, 求解了 TSPLIB 标准库中 54 个算例, 以验证 MVODM 的有效性. 求解结果表明: MVODM 能够非常有效地提高 4 种算法的求解质量, 部分改进后的算法质量甚至超过很多世界一流的构建型算法, 并且 MVODM 的执行效率非常高, 当算例的规模达到 2319 时在本实验计算机上的耗时仅为 0.076 s, 算法的耗时增加量几乎可以忽略不计.

关键词 组合优化 旅行商问题 构建型算法 计算方法 通用策略

1 引言

TSP 问题是由美国学者 Dantzig [1] 在 1954 年提出的. TSP 问题为典型的容易描述难以求解的 NP-hard 组合优化问题 [2]. 由于求解 TSP 问题的方法理论可以广泛应用于物资分配、生产调度、计算机网络、通信调度、机器人控制、电路板钻孔、X 光设备定位、VLSI 芯片设计及物流运输路线优化等现实问题, 所以半个多世纪以来有众多学者将 TSP 问题的求解算法作为研究课题, 从而产生了大量有关 TSP 问题的文献 [3~5].

研究 TSP 问题的文献主要集中于研究高效的启发式算法, 启发式算法分为构建型算法 (construction algorithms) 和改进型算法 [2] (improved algorithms). 构建型算法是从最初的空解开始, 迭代地添加解成分, 直到构建一个完整解. 经典的构建型算法有最近邻域算法、贪婪算法、插入算法和节约算法等 [2], 构建型算法的优点是算法规则比较简洁, 求解速度非常快, 缺点是求解质量不是非常精确. 改进型算法是在构建型算法构造出的解基础上, 采用优化法则进一步改进解质量的算法. 比较常见的改

引用格式: 饶卫振, 王新华, 金淳, 等. 一类求解 TSP 构建型算法的通用改进策略. 中国科学: 信息科学, 2015, 45: 1060–1079, doi: 10.1360/N112014-00144

进型算法有蚁群算法 [6,7]、模拟退火法 [8]、禁忌搜索法 [9,10]、遗传算法 [11]、粒子群算法 [12] 等. 这类算法的优点是求解质量非常高, 缺点是算法规则复杂, 求解速度比较慢.

由上述构建型算法和改进型算法的优缺点可知, 如果在保证算法优点的同时进一步克服算法的缺点将会提高算法的性能. 换句话说, 提高构建型算法性能的策略为: 在几乎不增加构建型算法耗时的同时, 显著提高构建型算法的求解质量; 提高改进型算法性能的策略为: 在几乎不损失改进型算法求解质量的前提下, 显著提高改进型算法的执行效率. 并且 Toth 和 Vigo [13] 指出, 因为经典构建型算法的规则简洁明了, 要提高算法的求解质量一般会使算法的规则复杂化, 从而在提高求解质量的同时也会显著增加其运算时间, 所以提高构建型算法性能的难度很大. 当前研究提高改进型启发式算法的文献相对较多, 如江贺等 [14] 提出了改进 LKH 的方法策略, 其改进效果明显有效. 然而, 由于构建型算法规则简洁, 执行效率高, 在要求快速响应的实际应用环境下具有重要的应用价值, 比如动态车辆路径问题 [15] 或随机 TSP 问题 [4] 等, 因此研究进一步提高构建型算法求解质量, 无论在算法设计的理论方面还是实践方面, 均具有重要的研究意义. 当前对构建型算法的改进研究主要有两种方式: (1) 改变算法本身的规则, 基于某个经典算法提出其改进版本, 如李随成等 [16] 基于最近邻域算法提出的两端延伸最近城市搜索法, 随后本文作者基于该算法又提出了双向扩展差额算法 [17]; (2) 研究多个算法的特点, 寻求能够优缺点互补的算法, 然后将多个算法结合得到混合算法, 如 Hassin 等 [18] 在插入算法和贪婪算法的基础上提出了该两者的混合算法.

本文基于 4 种具有贪婪属性的构建型算法: 最近邻域算法、两端延伸最近城市搜索法、双向扩展差额算法和贪婪算法, 分析其求解的缺陷, 借助 Held 和 Karp [19] 在 1970 年提出的一个改造距离矩阵模型, 提出了最小化 TSP 问题算例距离矩阵方差的方法 (minimizing variance of distance matrix, MVODM), 该方法能够有效地克服这 4 种算法的共同缺陷, 提高其求解质量. 值得指出的是本文提出的 MVODM 策略不需要改变算法本身的规则, 只需要变动 TSP 问题算例的距离矩阵, 这为改进算法性能提供了新的思路. 最后通过求解 54 个标准 TSP 算例验证 MVODM 提高算法求解质量的效果, 以及其执行的效率.

2 TSP 问题的描述

TSP 问题: 一个推销员要旅行 n 个城市, 且任意两个城市之间距离已知, 求推销员旅行每个城市一次且仅一次, 并回到起点城市的最短路径. 该问题可用数学符号表述为: 设 n 个城市之间的距离矩阵为

$$D=(d_{ij})_{n\times n}\begin{cases} d_{ij}\ \ (i\neq j),\\ M\ \ (i=j), \end{cases} \tag{1}$$

其中, M 为足够大正数. $n(n\geqslant 3)$ 个城市分别标号为 $\{1,2,3,\ldots,n\}$, 一个 TSP 问题的解就是 n 个城市标号的环状排列. 令 S 为存放了一个解的一维数组, 那么此解的旅行总距离 (路径总长度) 可由式 (2) 计算:

$$L(S)=\left(\sum_{i=0}^{n-2} d_{S[i]S[i+1]}\right)+d_{S[n-1]S[0]}. \tag{2}$$

显然, 使得 $L(S)$ 取得最小值的一维数组 S 中存放的排列为 TSP 问题的最优解 [2]. TSP 问题主要有 3 种分类方式 [20]: (1) 按距离矩阵 D 是否为对称矩阵, 分为对称 TSP(STSP) 问题和非对称

TSP(ATSP) 问题, 本文在未加说明情况下 TSP 含义均为对称 TSP; (2) 按任意 3 个城市间距离值是否满足三角不等式, 分为三角不等式 TSP 问题和非三角不等式 TSP 问题; (3) 按 n 个城市的坐标位置之间距离是否完全符合 Euclid 平面规则, 即任意两个城市之间距离 $d_{ij}=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$ (其中 (x_i,y_i) 表示顾客 i 的位置), 分为 Euclid TSP 问题和非 Euclid TSP 问题. 显然, Euclid TSP 问题必为对称三角不等式 TSP 问题, 反之则不成立.

边的含义: 将 n 个城市看作节点, 任意两个不同城市 i,j 之间的连线称为边 (i,j), 其边长为 d_{ij} (距离矩阵 D 中第 i 行 j 列元素), 其中 i,j 称为边 (i,j) 的两个端点, n 个节点的全图中共有 $n(n-1)/2$ 条边, 一个解 (路径) 中共有 n 条边.

3 4 种贪婪构建型算法及其特点分析

最近邻域算法 (the nearest neighborhood algorithm, NN) 的求解思路是以某城市为起点, 构造路径时每一次都选择最近的未旅行的城市作为下一个旅行城市, 直到旅行完所有的城市再回到起点城市, 得到一个解 [2].

两端延伸最近城市搜索法 (both ends extending and nearest city searching algorithm, BENCS) 是在 NN 算法基础上的改进算法, 其思路是首先以距离矩阵 D 中最短的边为起始路径; 然后分别判断该路径两端点的最邻近城市, 选择与最邻近城市距离更短的一端延伸扩展当前的路径, 直到最后形成一个连接了所有城市的圈 [16].

双向扩展差额算法 (two directions moving with difference algorithm, TDMDA) 是 NN 和 BENCS 的改进算法. TDMDA 求解 TSP 问题与 BENCS 的主要区别是每次选择延伸方向的规则不同, TDMDA 每次均计算路径端点与次近城市和最近城市的距离之差, 选择该差额大的端点作为延伸端 [17].

经典贪婪算法 (greedy algorithm, GR) 的求解原理是在距离矩阵中按规则依次添加当前最短边, 直到添加完 n 条边形成一个简单圈为止 [21]. 由于 TSP 问题是求长度最短的简单圈, 因此 GR 添加边的规则为, 每次在可添加的边 (未添加过的, 且不会使当前路径中出现连接三条边的点或出现少于 n 个点组成的简单圈) 中选择最短的边添加至当前图中, 最后添加完 n 条边形成一个 TSP 可行解 [21,22].

该 4 种算法的共同特征是均含有贪婪的特征, 即将 TSP 可行解视为由城市点之间连接的边构成的集合, 通过构造规则不断依次添加边的方式形成最终解. 为了更加直观地观察这 4 种算法的求解效果, 在此分别采用 NN, BENCS, TDMDA 和 GR 求解了 TSP 标准算例 kroa100, 求解后的结果如图 1 所示.

由图 1 可知, 这 4 种算法求解的结果均包含了部分明显不合理的长边, 并且通过分析发现这些较长边均是求解过程中后期添加的边. 其主要原因是, 在构造的初期以贪婪的方式选择能添加的最短边, 但在构造的后期, 由于算法规则必须要求形成一个可行解, 且可选择的边越来越少, 导致当前可添加的最短边, 从整个距离矩阵中来看也属于比较长的边. 总之, 这 4 种算法由于构造初期的贪婪, 以致构造后期付出添加长边的代价, 从而使整个解的质量下降. 所以, 如何适当的降低添加初期的贪婪性, 减少构造后期添加长边的概率, 是提高这 4 种算法求解质量的关键所在.

4 MVODM 距离矩阵改造法

Held Karp 模型是使用一个实数向量 $\boldsymbol{\pi}=(\pi_1,\pi_2,\ldots,\pi_n)$ 改造 TSP 问题距离矩阵的模型 [19], 如式 (3) 所示:

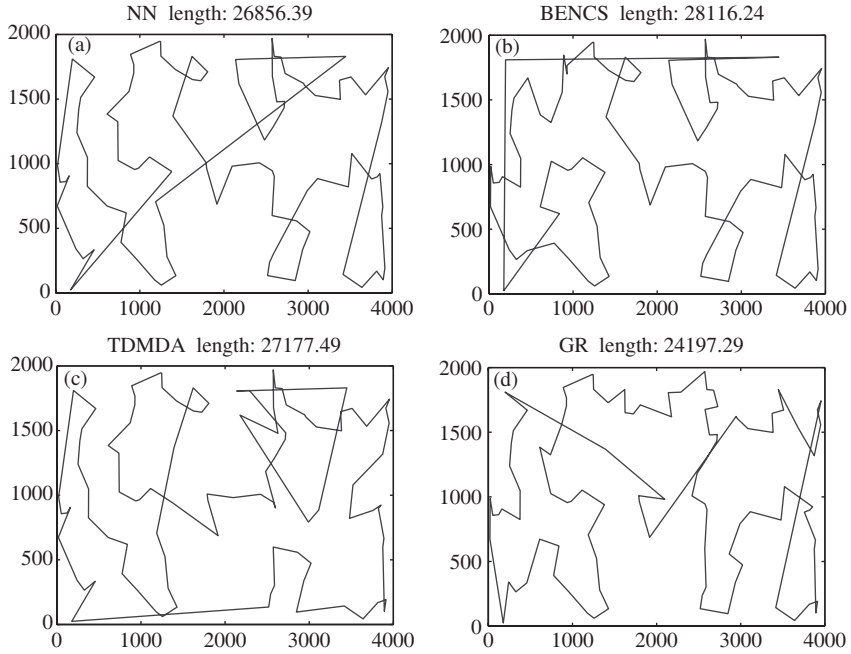


图 1 NN, BENCS, TDMDA 和 GR 求解 TSP 标准算例 kroa100 的结果

Figure 1 The solutions of NN, BENCS, TDMDA and GR for TSP benchmark instance kroa100

$$D=(d_{ij})_{n\times n}\begin{cases} D'=(d'_{ij})_{n\times n}\begin{cases} d'_{ij}=d_{ij}-\pi_i-\pi_j & (i\neq j),\\ d'_{ij}=M & (i=j),\end{cases}\\ \pi_i\in R \qquad (1\leqslant i\leqslant n),\end{cases} \tag{3}$$

设 S 为存放了一个 TSP 问题解的一维数组, $L(S)$ 表示该解按原距离矩阵计算的路径总长度, $L'(S)$ 表示该解按改造后距离矩阵计算的路径总长度. Held 和 Karp [19] 通过研究发现有定理 1 成立.

定理 1 [19] 任意 TSP 算例, 按照式 (2) 选择任意实数向量 $\boldsymbol{\pi}$ 改变其距离矩阵, 算例的最优解按照改变后距离矩阵计算路径长度仍然会是最优解.

证明 因 TSP 问题的解是 n 个城市标号的环状排列, 在解 (路径) 中每个城市均为两条边的端点, 因此据式 (2) 和 (3) 可得

$$\begin{aligned} L'(S) &= \sum_{i=0}^{n-2}\left(d'_{s[i]s[i+1]}\right)+d'_{s[n-1]s[0]}\\ &= \sum_{i=0}^{n-2}(d_{s[i]s[i+1]}-\pi_{s[i]}-\pi_{s[i+1]})+(d_{s[n-1]s[0]}-\pi_{s[n-1]}-\pi_{s[0]})\\ &= L(S)-2\sum_{i=1}^{n}\pi_i. \end{aligned} \tag{4}$$

由式 (4) 易见 $L(S)$ 与 $L'(S)$ 的大小关系. 显然, 对于任意实数向量 $\boldsymbol{\pi}$, 其分量累加和的两倍将是一个具体数值 (可能是负数), 那么算例解空间中每个解按照改变后距离矩阵计算的路径长度, 较原来均减小了这个具体数值. 因此, 由于路径长度都增加或减少相同的值不会影响解之间路径长度的比较, 所

以改造前后的距离矩阵具有同样的最优解. 证毕.

Held 和 Karp [19,23] 提出该转变距离矩阵的目的是: 求解一个实数向量 $\boldsymbol{\pi}^\star = (\pi_1^\star, \pi_2^\star, \ldots, \pi_n^\star)$, 使得最小 1-tree 中的边长之和最大, 由于 1-tree 中的边长之和是 TSP 问题最优解的下界, 也就是提高最优解下界的质量, 但由于该方法要不断的求解 1-tree, 其求解 $\boldsymbol{\pi}^\star$ 本身就是优化问题, 其计算量非常大.

本文思想: 由定理 1 可知, 任意实数向量 $\boldsymbol{\pi}$ 按 Held Karp 模型改造后的距离矩阵不会改变 TSP 问题最优解, 但是由于改造后的距离矩阵中的元素大小次序发生改变, 会使上述 4 种贪婪构建型算法的求解结果发生变化, 且不同实数向量 $\boldsymbol{\pi}$ 会导致不同的求解结果. 那么是否能够找到一个实数向量 $\boldsymbol{\pi}$, 使改造后的距离矩阵更加适宜该 4 种贪婪构建型算法求解? 4 种算法的主要缺点是在构造的后阶段要添加较长的边作为路径的组成部分, 由此产生质量低劣的最终解. 如果能使距离矩阵中的元素值 (边长) 之间更均匀, 减少距离矩阵中极大值的数量和大小, 就可能降低 4 种算法在后期构造质量恶化的可能性和程度, 从而提高该类算法的求解质量.

为了达到这一目的, 通过证明发现, 有定理 2 成立.

定理 2 存在一个实数向量 $\boldsymbol{\pi}^\star = (\pi_1^\star, \pi_2^\star, \ldots, \pi_n^\star)$, 使按 Held Karp 模型改造后的距离矩阵的方差最小.

证明 证明过程见附录 A. 证明思路是以改造后距离矩阵中元素的方差为目标函数, 以向量 $\boldsymbol{\pi}$ 的 n 个分量作为多元变量, 证明该多元函数存在全局极小值, 并求得 $\boldsymbol{\pi}^\star$. 通过证明发现多元函数具有无数个极小值, 且所有这些极小值取值相等, 显然, 其中任何一个极小值均为全局最小值. 即对应无数个 $\boldsymbol{\pi}^\star$(见附录 A 式 (A14)), $\boldsymbol{\pi}^\star$ 的计算方法如式 (5) 所示:

$$\begin{cases} \boldsymbol{\pi}^\star = (\pi_1^\star, \pi_2^\star, \ldots, \pi_n^\star), \\ \pi_k^\star = \frac{sumrow_k - (n-1)\bar{d}}{n-2} = \frac{sumrow_k - \overline{sumrow}}{n-2} (1 \leqslant k \leqslant n), \\ \bar{d} = \frac{\sum_{i=2}^{n} \sum_{j=1}^{i-1} d_{ij}}{n(n-1)/2}, sumrow_k = \sum_{j=1}^{n, j \neq k} d_{kj}, \overline{sumrow} = \frac{\sum_{k=1}^{n} sumrow_k}{n}, \end{cases} \tag{5}$$

其中, $sumrow_k$ 为距离矩阵 D 中第 k 行元素 (对角线上元素除外) 之和; $\overline{sumrow}$ 为 n 个 $sumrow_k$ 的平均值, $\bar{d}$ 为对称矩阵 D 中除对角线上为 M 元素之外的所有元素的平均值.

由于一组数与对应平均值的离差之和为零, 因此, 式 (5) 中 $\sum \pi_k^\star = 0$. 由此根据式 (4) 可知, TSP 的任何解的路径长度按改造前后的距离矩阵计算具有相同的值, 即任何解按照改造前和改造后的距离矩阵计算的总路径长度肯定一致. 这种改造距离矩阵的方法就称为 —— 距离矩阵方差最小法 (minimizing variance of distance matrix, MVODM).

5 算例求解

5.1 MVODM 的效果

本小节主要目的是验证 MVODM 策略提高 4 种算法 NN, BENCS, TDMDA 和 GR 求解质量的效果. 为此, 采用上述 4 种算法在不结合和结合 MVODM 策略两种情况下, 求解了 TSPLIB 标准库中的 54 个 Euclid TSP 算例[1)], 且算例的城市数量范围从 16 至 2319 充分考虑了算例的不同规模级别. 本

1) http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/.

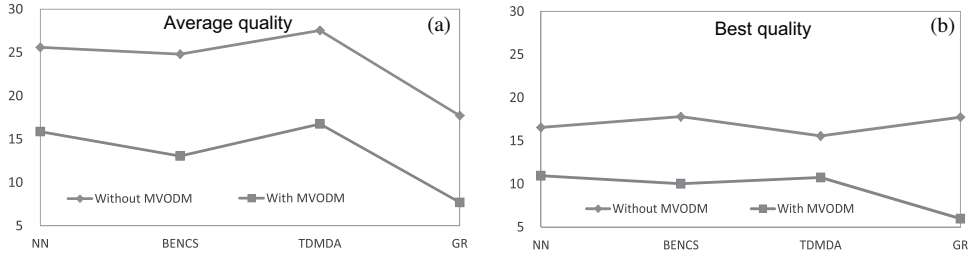


图 2 4 种算法在结合 MVODM 策略前后求解质量对比图

Figure 2 The solution quality comparison of 4 algorithms with and without MVODM

次算例求解环境为: Studio Visual C++6.0, C 语言编程; 运行平台: CPU 为 Inter(R)core(TM) 2 Duo, 内存为 2.0 G, 主频为 2.93 GHz. NN, BENCS, TDMDA 和 GR 在不结合 MVODM 策略的情况下求解 54 个算例的结果如表 1 所示, 该 4 种算法在结合 MVODM 策略后求解结果如表 2 所示.

表 1 和 2 关键字说明, "算例名称", 算例名称中包含的数字为算例的城市数量; "最优路径长度", 算例的已知最优解路径长度; "平均解, 最优解", 分别代表算法求解的平均求解质量和最优求解质量, 求解质量用百分数表示, 计算方法为 100%×(解路径长度 − 最优路径长度)/最优路径长度. 需要说明的是 NN, BENCS 和 TDMDA 由于求解结果与初始选择的起点或起始边相关, 为不确定性算法, 在此, 该 3 种算法根据算例规模 n 不同, 对不同算例求解的解数量不同, 规则如下: 当 $16 \leqslant n < 600$, 求解数量为 n; $600 \leqslant n < 1000$, 求解数量为 200; $1000 \leqslant n < 2000$, 求解数量为 100; $n \geqslant 2000$, 求解数量为 50. 故该 3 种算法求解算例结果存在平均解质量和最优解质量. 然而, GR 为确定性算法, 即每次求解的解均相同, 因此 GR 求解任何算例的解数量均为 1, 也就是说只有唯一求解结果.

如表 1 所示, 对于所有 54 个算例, NN, BENCS 和 TDMDA 的平均求解质量的平均值均在 25% 左右, 最优求解质量平均值在 16% 左右, GR 的求解质量的平均值为 17.73% 接近该 3 种算法的最优求解质量. 然而, 在结合 MVODM 策略后 (如表 2 所示), 该 4 种算法的求解质量有了非常显著的提高, NN 算法求解所有算例的平均解和最优解的平均值质量, 分别从 25.60% 和 16.54% 提高至 15.87% 和 10.95%, BENCS 和 TDMDA 也具有和 NN 类似幅度的提高, GR 的求解质量由 17.73% 直接提高至 7.71%. 为了更加直观地展示 MVODM 提高该 4 种算法求解质量的效果, 在此用折线图对比了 4 种算法结合 MVODM 前后的求解质量, 如图 2 所示.

由图 2(a) 和 (b) 可知, MVODM 对提高 4 种算法 NN, BENCS, TDMDA 和 GR 的求解质量具有显著效果, 且对前 3 种算法提高平均解质量的幅度要大于最优解, 也就是说, MVODM 在提高此 3 种算法质量的同时, 也提高了算法的鲁棒性. 另外, MVODM 与 GR 结合后较前 3 种算法具有更好的求解质量.

求解 TSP 问题的构建型算法, 当求解质量在 15% 以下时被认为是优秀水平, 当求解质量能够达到 10% 左右时可称为一流构建型算法 [24]. 显然, 在结合 MVODM 策略之前, NN, BENCS, TDMDA 和 GR 不属于优秀构建型算法, 然而在结合 MVODM 策略后前 3 种算法达到优秀构建型算法水平, 而 GR 则达到一流构建型算法水平. Johnson [24] 通过求解一个包含 10000 个城市的标准算例, 以评估当前经典构建型算法的求解质量, 在此用改进后的 4 种算法求解了该算例, 以比较改进后算法的求解质量水平, 其比较结果如表 3 所示, 其中 NN, BENCS, TDMDA 结合 MVODM 只求解该算例一次.

由表 3 可知, MVODM 结合 4 种算法的求解质量优于大部分经典构建型算法, 其中 GR+MVODM 的求解质量仅略次于 Christo-S. 然而 Christo-S 的耗时是 Greedy(GR) 算法的 4∼6 倍 [24].

表 1　NN, BENCS, TDMDA 和 GR 求解 54 个算例结果

Table 1　Experiment results for NN, BENCS, TDMDA and GR for 54 TSP benchmark instances

No.	Instance	Optimal length	NN		BENCS		TDMDA		GR
			Average	Best	Average	Best	Average	Best	
1	ulysses16	68.59	31.14	12.45	25.70	22.86	38.84	**8.11**	26.53
2	ulysses22	70.13	30.36	23.93	25.85	23.68	35.78	**9.03**	27.53
3	bayg29	9074.15	21.72	9.82	25.87	14.24	24.17	12.44	**8.95**
4	bays29	2020	16.38	**5.64**	16.24	12.57	20.17	9.46	12.13
5	dantzig42	699	28.12	17.61	25.42	19.83	33.69	**11.85**	20.71
6	swiss42	1273	25.39	12.88	28.27	23.41	31.98	**12.25**	23.10
7	att48	33523.71	24.44	17.04	26.90	19.52	21.31	**13.08**	19.80
8	eil51	426	32.18	18.73	29.07	20.04	32.58	17.77	**13.03**
9	berlin52	7542	24.28	**8.49**	23.09	13.27	35.57	14.31	31.98
10	st70	675	22.34	12.84	23.17	15.39	30.76	15.13	**10.52**
11	eil76	538	23.70	13.88	22.87	**10.38**	24.30	13.29	14.71
12	pr76	108159	36.08	**21.04**	36.53	28.45	38.49	23.79	29.76
13	gr96	512.31	27.86	17.76	23.73	18.24	31.13	18.22	**13.23**
14	rat99	1211	21.88	13.09	21.78	**12.70**	24.38	13.73	22.30
15	kroa100	21282	27.08	16.05	26.93	23.14	28.83	17.77	**13.70**
16	krob100	22141	26.04	16.90	21.66	15.88	20.73	**6.89**	16.59
17	kroc100	20749	26.40	13.58	25.91	12.78	18.98	**9.79**	12.94
18	krod100	21294	29.06	16.73	29.91	22.71	33.87	19.58	**14.82**
19	kroe100	22068	24.60	12.86	24.69	14.90	24.76	**9.64**	12.59
20	rd100	7910	27.91	19.18	24.46	18.05	30.99	24.00	**16.99**
21	eil101	629	26.42	**17.07**	28.58	18.24	30.33	17.36	20.80
22	lin105	14379	30.19	17.81	31.34	20.75	24.55	**13.64**	16.64
23	gr120	1666.51	24.95	11.03	23.23	12.28	22.57	**9.98**	19.72
24	pr124	59030	20.71	13.60	19.00	14.09	19.48	**4.66**	10.11
25	bier127	118282	23.58	13.26	21.14	14.86	26.29	14.18	**13.06**
26	ch130	6110.86	26.34	17.80	24.79	14.84	27.84	**15.03**	28.38
27	pr136	96772	25.76	18.38	24.11	18.73	27.10	**16.11**	19.93
28	gr137	698.53	47.67	33.16	40.37	30.13	38.95	26.64	**21.18**
29	pr144	58537	12.59	4.14	10.00	6.84	14.41	**3.21**	12.49
30	kroa150	26524	26.83	18.69	26.65	21.01	29.24	**17.82**	20.24
31	krob150	26130	31.90	19.86	27.88	21.50	32.13	**15.65**	20.25
32	u159	42080	28.67	**15.46**	25.65	19.28	30.94	19.75	16.24
33	rat195	2323	19.66	13.15	18.51	12.91	19.16	**10.46**	18.73
34	kroa200	29368	27.58	**17.64**	33.02	21.74	28.94	20.26	17.83
35	krob200	29437	25.58	20.24	26.98	20.81	30.34	**17.86**	22.24
36	ts225	126643	17.33	10.93	16.25	9.87	16.11	7.08	**5.19**
37	tsp225	3916	25.63	**18.31**	27.30	18.39	33.81	18.33	19.58
38	pr226	80369	24.20	**15.16**	22.78	16.89	32.80	17.62	20.69

续表 1

No.	Instance	Optimal length	NN		BENCS		TDMDA		GR
			Average	Best	Average	Best	Average	Best	
39	gil262	2378	29.75	21.27	27.92	18.91	25.88	17.41	**13.21**
40	pr299	48191	30.59	20.95	32.48	**20.19**	35.60	21.23	19.59
41	lin318	42029	25.15	**17.10**	22.89	17.23	24.76	18.35	18.75
42	rd400	15281	25.77	19.78	24.98	19.76	29.23	18.48	**17.10**
43	pr439	107217	26.56	18.66	23.47	16.47	28.64	16.38	**16.16**
44	pcb442	50778	22.55	16.10	21.07	**15.18**	25.73	17.50	19.41
45	d493	35002	25.23	18.56	22.26	17.48	27.68	19.20	**16.72**
46	ali535	2023.39	31.76	20.29	33.03	24.18	34.22	**18.70**	18.89
47	u574	36905	27.78	20.62	26.74	21.10	29.41	**19.06**	21.72
48	rat575	6773	22.51	17.71	23.27	**17.65**	26.08	18.19	21.40
49	dsj1000	18659688	28.99	23.66	28.52	21.49	31.39	22.83	**16.32**
50	pcb1173	56892	26.33	22.50	27.95	23.53	29.34	23.13	**17.23**
51	vm1748	336556	26.16	20.65	27.31	20.48	28.34	22.19	**17.40**
52	d2103	80450	12.33	**8.72**	14.35	8.39	14.31	10.86	13.33
53	u2152	64253	23.12	21.95	21.29	18.13	26.19	19.84	**15.78**
54	u2319	234256	18.60	16.19	16.94	14.87	17.95	15.58	**13.35**
Average			25.60	16.54	24.81	17.79	27.55	**15.56**	17.73

表 2 NN, BENCS, TDMDA 和 GR 结合 MVODM 求解 54 个算例结果

Table 2 Experiment results for NN, BENCS, TDMDA and GR with MVODM for 54 TSP benchmark instances

No.	Instance	Optimal length	NN+MVODM		BENCS+MVODMS		TDMDA+MVODM		GR+ MVODM
			Average	Best	Average	Best	Average	Best	
1	ulysses16	68.59	12.67	9.26	10.96	9.26	10.34	**8.72**	10.50
2	ulysses22	70.13	13.38	9.11	11.36	10.62	11.99	10.05	**8.67**
3	bayg29	9074.15	9.26	2.86	6.73	**1.55**	7.30	2.16	5.07
4	bays29	2020	13.88	7.52	12.84	7.08	8.46	**5.10**	5.89
5	dantzig42	699	9.48	0.16	6.20	0.64	9.78	0.52	**0.11**
6	swiss42	1273	15.64	10.21	11.62	9.19	14.99	10.21	**9.11**
7	att48	33523.71	10.26	**3.41**	8.56	5.42	17.07	9.76	7.99
8	eil51	426	13.90	7.27	10.01	3.64	10.04	4.49	**0.83**
9	berlin52	7542	16.91	11.95	14.18	9.93	15.42	**5.45**	9.73
10	st70	675	12.13	**5.67**	10.11	5.79	20.12	6.53	7.48
11	eil76	538	15.04	10.74	11.91	9.65	11.97	7.99	**7.04**
12	pr76	108159	12.29	7.98	13.85	7.56	13.43	8.08	**3.51**
13	gr96	512.31	16.88	10.72	12.28	8.78	15.06	8.25	**3.43**
14	rat99	1211	15.70	10.30	12.83	**8.25**	13.24	8.74	8.49
15	kroa100	21282	15.29	9.93	10.91	7.97	23.37	13.17	**5.76**
16	krob100	22141	18.51	**9.58**	13.35	10.76	18.48	10.54	11.57

续表 2

No.	Instance	Optimal length	NN+MVODM		BENCS+MVODMS		TDMDA+MVODM		GR+MVODM
			Average	Best	Average	Best	Average	Best	
17	kroc100	20749	18.40	**11.20**	16.84	11.79	19.66	12.42	11.41
18	krod100	21294	16.41	10.15	12.43	**7.69**	21.98	12.65	8.44
19	kroe100	22068	17.28	10.33	13.65	8.23	18.65	15.27	**5.60**
20	rd100	7910	16.68	11.90	12.52	8.89	16.44	9.57	**7.78**
21	eil101	629	14.10	10.62	13.14	10.25	15.76	**7.84**	10.43
22	lin105	14379	16.51	8.43	12.79	10.44	19.06	6.68	**5.25**
23	gr120	1666.51	11.03	6.07	8.09	6.46	10.74	7.04	**3.61**
24	pr124	59030	21.24	12.39	15.42	13.00	22.78	13.57	**2.10**
25	bier127	118282	14.11	8.84	12.87	7.63	16.46	11.68	**6.52**
26	ch130	6110.86	14.99	10.33	11.91	9.05	17.79	12.14	**7.32**
27	pr136	96772	12.96	9.07	12.51	**8.20**	13.91	10.01	9.14
28	gr137	698.53	17.73	12.52	12.09	**8.19**	20.67	12.64	11.12
29	pr144	58537	7.79	1.29	3.14	0.88	8.57	**0.62**	4.17
30	kroa150	26524	17.88	13.12	14.19	11.41	22.54	14.48	**4.35**
31	krob150	26130	13.46	9.88	12.26	8.45	20.47	12.56	**7.99**
32	u159	42080	25.02	12.03	16.20	12.99	18.18	11.77	**7.95**
33	rat195	2323	9.64	**4.60**	8.72	5.36	11.68	7.12	5.42
34	kroa200	29368	19.30	15.49	16.67	13.73	23.36	14.51	**6.81**
35	krob200	29437	18.82	12.75	13.61	11.26	17.46	**10.29**	12.67
36	ts225	126643	7.59	5.19	4.26	**2.90**	4.76	2.77	11.28
37	tsp225	3916	16.89	11.48	13.91	9.56	16.72	8.95	**6.49**
38	pr226	80369	16.10	**7.16**	12.08	8.32	22.38	11.47	7.21
39	gil262	2378	19.80	16.73	16.42	14.40	19.72	13.50	**10.05**
40	pr299	48191	22.46	17.12	17.05	15.16	24.31	16.20	**9.38**
41	lin318	42029	15.63	11.42	12.05	10.32	17.10	11.18	**7.16**
42	rd400	15281	16.69	13.33	14.44	12.66	19.09	13.37	**7.13**
43	pr439	107217	22.34	17.47	20.59	17.64	25.57	17.64	**11.73**
44	pcb442	50778	17.92	14.93	14.53	12.89	16.92	13.03	**10.20**
45	d493	35002	17.60	15.53	15.12	13.39	17.34	13.26	**10.67**
46	ali535	2023.39	20.28	15.58	17.73	12.90	22.52	14.13	**10.15**
47	u574	36905	19.49	15.92	17.69	14.20	20.55	15.00	**7.60**
48	rat575	6773	15.10	11.19	12.52	9.68	16.55	12.36	**7.18**
49	dsj1000	18659688	21.67	19.09	21.11	20.68	26.09	21.81	**12.25**
50	pcb1173	56892	21.48	19.89	18.33	17.20	20.20	18.53	**11.89**
51	vm1748	336556	21.83	19.22	20.42	20.28	22.71	18.42	**10.90**
52	d2103	80450	12.23	11.49	11.74	10.71	12.57	10.57	**7.28**
53	u2152	64253	23.24	22.92	23.24	22.92	20.01	19.18	**9.74**
54	u2319	234256	7.77	7.46	6.20	5.93	6.47	6.11	**4.54**
Average			15.87	10.95	13.05	10.02	16.75	10.74	**7.71**

表 3 4 种改进后算法与当前经典构建型算法结果比较

Table 3 Comparison of 4 improved algorithms and classic construction algorithms

Algorithms	Quality (%)	Algorithms	Quality (%)
Spacefill	34.56	FI	13.35
Strip	30.75	AppChristo	11.05
Karp-20	29.34	Christo-S	9.81
NI	26.50	NN+MVODM	13.18
NN	24.79	BENCS+MVODM	13.54
CHCI	20.73	TDMDA+MVODM	12.02
Greedy	16.42	GR+MVODM	10.15

表 4 GR+MVODM 与 CIR 算法结果比较

Table 4 Comparison of GR+MVODM and CIR algorithm

Instance	CIR (%)	GR+MVODM (%)	Instance	CIR (%)	GR+MVODM (%)
krod100	15	**8.44**	kroa150	10	**4.35**
kroe100	9	**5.60**	krob150	17	**7.99**
rd100	10	**7.78**	u159	11	**7.95**
eil101	**10**	10.43	rat195	13	**5.42**
lin105	11	**5.25**	kroa200	12	**6.81**
gr120	10	**3.61**	krob200	16	**12.67**
pr124	3	**2.10**	ts225	14	**11.28**
bier127	15	**6.52**	tsp225	10	**6.49**
ch130	10	**7.32**	vm1748	15	**10.90**
pr136	14	**9.14**	d2103	**5**	7.28
gr137	15	**11.12**	u2152	12	**9.74**
pr144	5	**4.17**	u2319	**4**	4.54

另外, Hassin [18] 在 2008 年提出了优秀的构建型算法 CIR. 在此比较了 CIR 和 GR+MVODM 的求解质量, 其比较结果如表 4 所示, 其中 CIR 的求解质量在文献 [18] 中只保留了整数. 通过比较该两种算法求解的 24 个算例结果可知, 仅有 3 个算例 eil101, d2103 和 u2319, 算法 CIR 略微优于 GR+MVODM, 而其他 21 个算例 GR+MVODM 具有明显的优势. 并且 CIR 算法的求解耗时较最近插入算法 CI 增加了 70% 之多 [18]. 然而 CI 与表 3 中一般插入算法 NI 的求解耗时基本一致, 根据文献 [24] 可知 NI 耗时是 Greedy(GR) 耗时的 6∼8 倍.

为了进一步说明 GR+MVODM 性能, 在此比较了该算法与 TSP Challenge 网站中列出的优秀算法的性能. 由于 GR+MVODM 求解 d2103 算例的质量为 7.28%, 最接近其平均求解质量 7.71% (见表 2). 另外, TSP Challenge 中很多算法也求解了算例 d2103, 并且 TSP Challenge 组织者为了进行公平比较, 折算了各算法的标准化耗时. GR+MVODM 和 TSP Challenge 中具有代表性 (根据求解质量水平选择) 的 19 种算法求解 d2103 的质量和标准化耗时等信息, 如表 5 所示.

在表 5 中需要说明的是, (1) TSP Challenge 中的算法名称均为缩写, 详细请参考文献 [24]; (2) 表 5 中的 Greedy-JM.MIPS 算法即为本文中的 GR 算法, 由于 GR 与 GR+MVODM 求解 d2103 的耗时相近 (请见 5.2 小节), 因此, 表 5 中 GR+MVODM 的标准化耗时与 Greedy-JM.MIPS 算法相同, 等于

表 5 GR+MVODM 与 TSP Challenge 中的算法求解 d2103 的结果比较

Table 5 Results comparison of d2103 solving by GR+MVODM and algorithms of on TSP Challenge

No.	Instance	Quality (%)	Normalized time (s)	Category
1	ILK-JM-10N	0.00	1475.40	Improved
2	ILK-NYYY-10N	0.09	909.08	Improved
3	Tourmerge	0.32	124.53	Improved
4	VNS-3Hyper-1000	1.20	56.78	Improved
5	LK-Neto0-20n	2.03	10.78	Improved
6	3opt-ABCC	5.43	0.09	Improved
7	VRPR	6.85	1.01	Improved
8	2opt-JM-10	7.27	0.17	Improved
9	**GR+MVODM**	7.28	0.03	Construction
10	AppChristo-R	8.10	0.31	Construction
11	2.5opt-ABCC	8.62	0.30	Improved
12	2opt-ABCC	9.57	0.30	Improved
13	Q-Boruvka	9.62	0.03	Construction
14	Greedy-ABCC	10.52	0.03	Construction
15	Boruvka.Sparc	13.30	0.01	Construction
16	Boruvka.MIPS	13.41	0.03	Construction
17	Greedy-JM.MIPS	13.47	0.03	Construction
18	Greedy-JM.Xeon	13.47	0.03	Construction
19	Greedy-JM.Alpha	13.47	0.04	Construction
20	NN-ABCC	14.85	0.03	Construction

0.03; (3) 算法按求解质量从优到劣排序.

由表 5 可知, GR+MVODM 优于所有的构建型算法, 甚至优于两种改进型算法 (由于改进型算法是在某些构建型算法结果基础上进一步优化, 一般情况下, 改进型算法质量要优于构建型算法). 虽然部分改进型算法取得了非常好的求解结果, 甚至求得最优解, 但同时付出了巨大的耗时代价. 算法的主要性能一般通过求解质量和运算速度两个指标来体现, 然而, 由于运算平台的差异, 运算速度通常难以比较. 但在 TSP Challenge 中通过运行标准算法的方式很好地解决了该问题 [24]. 因此, 相对来说, TSP Challenge 中算法的运行速度的比较非常公平. 根据算法的求解质量和运行速度指标, 图 3 更加直观地比较了表 5 中 20 种算法的综合性能.

显然, 在图 3 中算法 (除 GR+MVODM 外, 其他算法用表 5 中的对应序号表示) 越靠近左下角表示又快又好. 一般情况是构建型算法 (图 3 中 ∘ 图标) 速度更快, 而改进型算法 (图 3 中 △ 图标) 求解质量更高. 但改进型算法在追求高精度求解质量时, 也会付出巨大的耗时代价. 如在表 5 中序号为 1 的算法 ILK-JM-10N 虽然求解到了 d2103 目前已知的最优解, 但耗时量相当于 GR+MVODM 的 49180 倍.

5.2 MVODM 的效率

如上所述, 算法的求解质量只是衡量算法优劣的指标之一, 算法的求解速度也是算法性能的主要指标. 由 5.1 小节的实例求解可知, MVODM 对提高本文提到的 4 种算法求解质量的效果明显, 那么

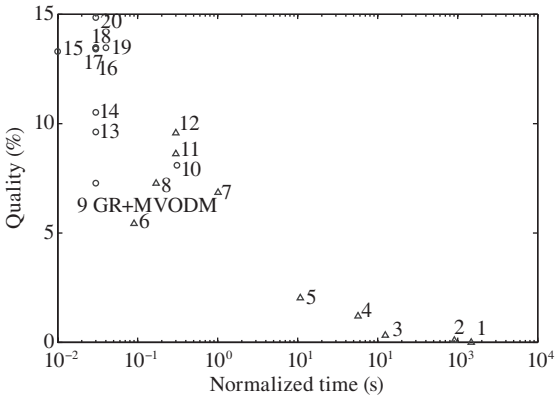


图 3 GR+MVODM 和 TSP Challenge 中 19 种算法的性能比较

Figure 3 The performance comparison of GR+MVODM and 19 algorithms on TSP Challenge

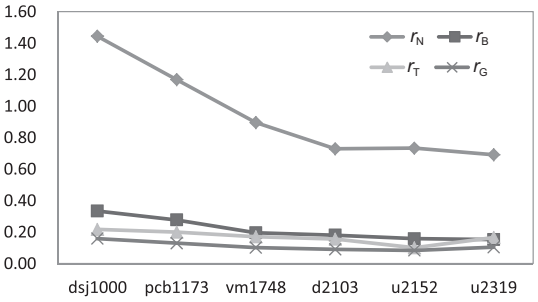


图 4 执行 MVODM 的耗时占 4 种算法求解耗时的比率

Figure 4 The ratios of MVODM running time to the entire running time for 4 algorithms

在此基础上是否要以牺牲算法的求解速度为代价?

为了展示 MVODM 的执行效率, 基于 6 个规模在 1000 以上的算例 (由于 MVODM 的求解速度太快, 小算例难以准确计时) 分别计算了 MVODM 的耗时与算法求解耗时的比率 $r(\%)$, 且 4 种算法的该比值分别用 r_{N}, r_{B}, r_{T} 和 r_{G} 表示, 具体数据如图 4 所示.

由图 4 可知, MVODM 的耗时占算法 NN 求解算例 1 次耗时的比率 r_{N} 在 1% 左右, 而对于后 3 种算法该指标 r_{B}, r_{T} 和 r_{G} 仅为 0.2% 左右, 并且随着算例规模的增加均有下降的趋势. 在本文的运行平台上采用 MVODM 转换规模为 2319 的 u2319 算例的耗时仅为 0.076 s. 需要指出的是, 对于不确定性算法 NN, BENCS 和 TDMDA 一般需要求解多个解, 然而在结合 MVODM 策略时, MVODM 仅需要执行一次. 因此, 该 3 种算法结合 MVODM 时, 其需要付出的耗时代价较图 4 要更少. 所以可以说, 该 4 种算法结合 MVODM 策略时, 其算法的耗时增加量非常小, 几乎可以忽略不计.

5.3 MVODM 的有效性分析

在第 3 节中分析 4 种算法的共同特点是, 由于构造初期的贪婪, 以致构造后期付出添加长边的代价, 从而使整个解的质量下降. 然后通过求解算例 kroa100 印证了该分析结论. 那么, MVODM 的有效性就是帮助该 4 种算法在构造初期适当降低其贪婪性, 尽量使其在后期添加的边不是太长, 从而提高整体解的质量. 下面同样以求解 kroa100 算例为例, 并分别采用 4 种算法 NN, BENCS, TDMDA 和 GR 结合 MVODM 进行求解, 其求解结果如图 5 所示.

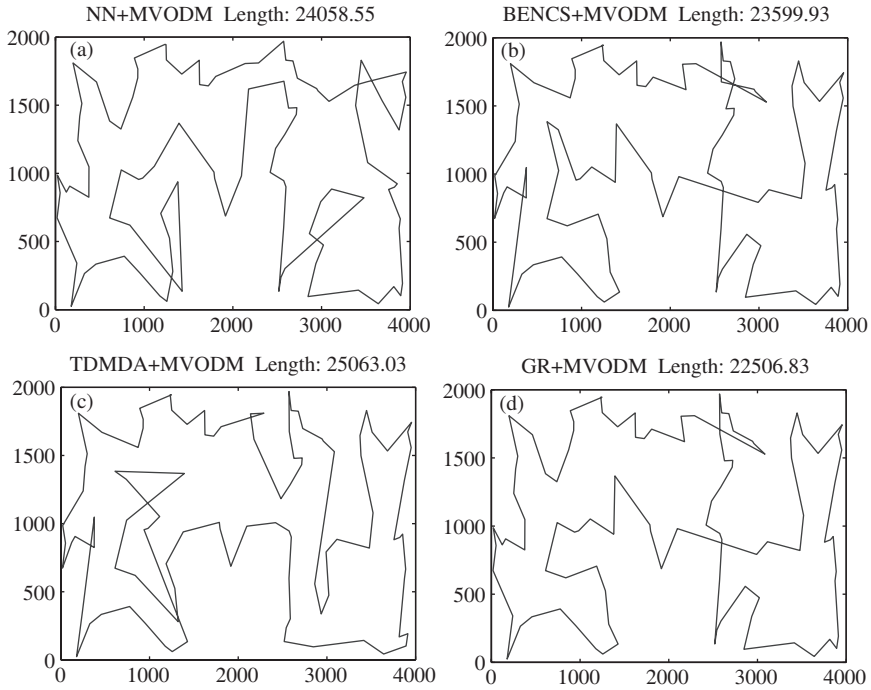


图 5 NN, BENCS, TDMDA 和 GR 结合 MVODM 求解 TSP 标准算例 kroa100 的结果

Figure 5 The solutions of NN, BENCS, TDMDA and GR with MVODM for TSP benchmark instance kroa100

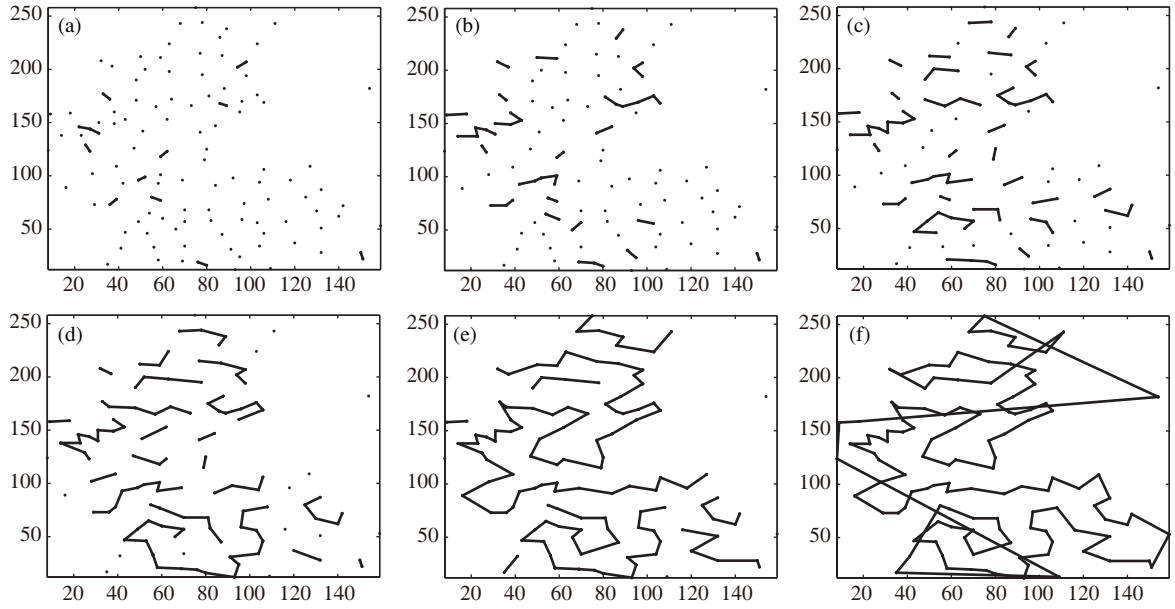


图 6 GR 求解算例 gr120 分别添加 12, 36, 60, 84, 108 和 120 条边后的解

Figure 6 The tours of GR solving instance gr120 after 12, 36, 60, 84, 108 and 120 edges have been added

由图 5 可知, 结合 MVODM 后的 4 种算法求解 kroa100 的结果中虽然也存在少数较长边, 但与图 1 中的结果相比, 无论是长边的长度还是数量均有明显减少.

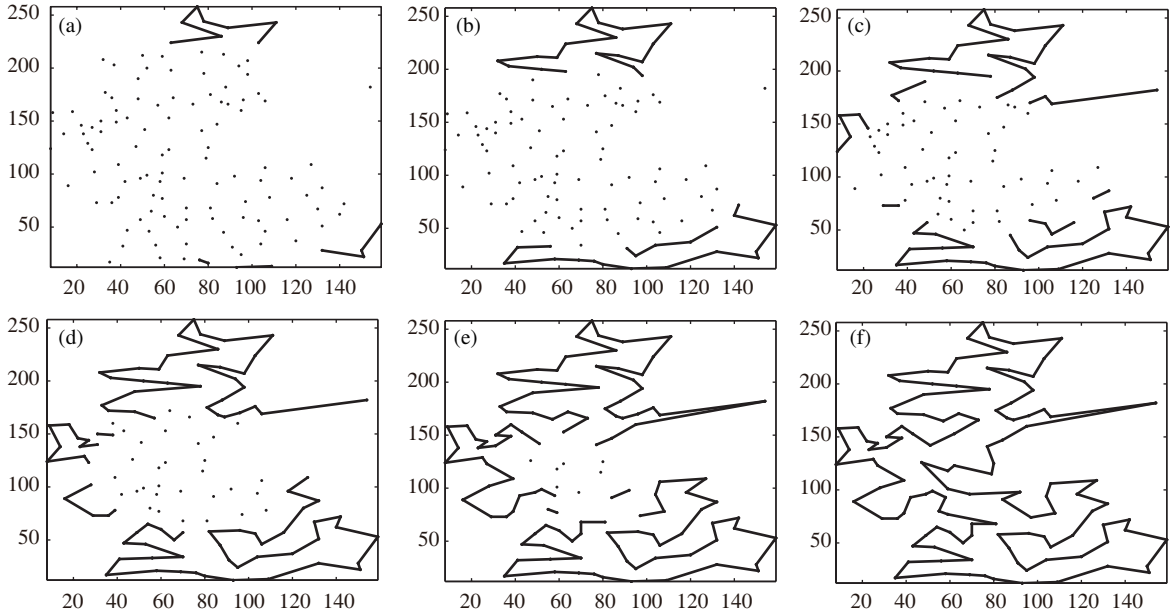


图 7 GR+MVODM 求解算例 gr120 分别添加 12, 36, 60, 84, 108 和 120 条边后的解

Figure 7 The tours of GR+MVODM solving instance gr120 after 12, 36, 60, 84, 108 and 120 edges have been added

为进一步分析 MVODM 提高贪婪构建型算法质量的内在原因, 在此以算例 gr120 为例, 在图 6 和 7 中分别记录了构建型算法 GR 和 GR+MVODM, 在分别添加完 12, 36, 60, 80, 108, 120 条边后的解路径.

通过对比图 6 和 7 可知, 图 6 中 GR 添加边的过程, 开始出现的边位置随机, 从而导致后期未连接两条边的节点位置比较分散, 为了形成一个 TSP 可行解 (每个点连接两条边, 且只有 1 个圈), 不得不连接一些长边, 最后导致解质量较差. 在图 7 中, MVODM 对距离矩阵中的数据进行了改变, 结合式 (3) 和 (5) 可知, 其改变方式相当于缩短了以边缘位置为端点的边长度, 延长了以中间位置节点为端点的边长度. 因此, GR+MVODM 优先添加边缘位置边, 这使得添加边的过程由外至内, 所以在后期未连接边的点基本分布在中间位置, 然后中间位置点之间的边长均较短, 从而克服了 GR 在后期添加较长边的缺点. 这也正是 MVODM 能够提高贪婪构建型算法质量的主要原因.

5.4 MVODM 的适用范围

通过上述实例求解与分析可得, MVODM 在不改变算法规则的情况下, 以非常微小的耗时代价, 能够有效地提高一类贪婪构建型算法的求解质量. 结合 MVODM 的贪婪构建型算法的主要应用包括:

(1) 为求解 TSP 问题的精确算法 (能够保证求解到最优解), 如分支定界法, 提供求解上界;

(2) 为改进型算法提供质量更为满意的初始优化解, 以节约总耗时甚至求解得到更优的最终解;

(3) 为动态性较强的问题快速提供质量满意的解, 如动态车辆路径问题, 要求算法能够在短时间之内结合新信息求解出满意方案 [25];

(4) 求解大规模车辆路径问题, 作者基于该思想已尝试求解大规模 CVRP 问题, 并取得了有效的结果 [26].

6 结论

本文针对 4 种传统构建型算法 NN, BENCS, TDMDA 和 GR, 通过分析发现构造初期的贪婪是其存在的共同缺陷; 基于该缺陷提出了转变算例距离矩阵的策略 MVODM, 然后通过求解标准算例验证了该策略的有效性, 本文主要结论如下:

(1) 4 种构建型算法 NN, BENCS, TDMDA 和 GR 的最大缺陷是, 在构造的初期以贪婪的方式构造解确实能够添加部分非常短的边, 但在构造的后期由于可选择边的数量非常有限, 就难以避免需要添加较长的边以形成可行解, 从而导致整体解的质量不高;

(2) TSP 算例的距离矩阵通过采用 Held Karp 模型变换, 可以得到无数个距离矩阵, 采用 MVODM 策略能够计算出一个 Held Karp 模型中的矢量 $\boldsymbol{\pi}^{\star}$, 而借助该矢量转变得到的距离矩阵的元素之间最均匀, 即方差最小;

(3) 通过求解 54 个标准 TSP 算例发现, MVODM 能够非常有效的提高 4 种构建型算法的求解质量, 使 NN, BENCS, TDMDA 达到优秀构建型算法水平, GR 甚至超过部分世界一流构建型算法和部分改进型算法;

(4) MVODM 执行效率非常高, 使 NN 算法耗时增加 1% 左右, 使 BENCS, TDMDA 和 GR 的耗时仅增加 0.2% 左右.

MVODM 策略只是改变了算法求解的距离矩阵, 没有改变算法任何求解规则, 从理论上来说可以与任何算法相结合, 在以后的研究中将对 MVODM 进一步深化, 以更大幅度地提高算法的质量或提高更多其他算法的求解质量.

致谢 感谢 Bell 实验室算法研究部的首席教授 Johnson S. David 给本研究提出的宝贵意见.

参考文献

1 Dantzig G B, Fulkerson D R, Johnson S M. Solution of a large scale traveling-salesman problem. Oper Res, 1954, 2: 393–410
2 Junger M, Reinelt G, Rinaldi G. Handbooks in OR&MS: Chapter4 the Traveling Salesman Problem. Holland: Elsevier Sci, 1995. 225–330
3 Martınez M A, Cordeau J F, Amico M A, et al. A branch-and-cut algorithm for the double traveling salesman problem with multiple stacks. Informs J Comput, 2013, 25: 41–55
4 Ghiani G, Manni E, Thomas B W. A comparison of anticipatory algorithms for the dynamic and stochastic traveling salesman problem. Transport Sci, 2012, 46: 374–387
5 Dash S, Günlük O, Lodi A, et al. A time bucket formulation for the traveling salesman problem with time windows. Informs J Comput, 2012, 24: 132–147
6 Dorigo M, Stutzle T. Ant Colony Optimization. Cambridge: MIT Press, 2004. 40–65
7 Ugur A, Aydin D. An interactive simulation and analysis software for solving TSP using ant colony optimization algorithms. Adv Eng softw, 2009, 40: 341–349
8 Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing. Science, 1983, 220: 671–680
9 Glover F. Tabu search — Part I. ORSA J Comput, 1989, 1: 190–206
10 Glover F. Tabu search — Part II. ORSA J Comput, 1990, 2: 4–32
11 Nagata Y, Kobayashi S. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. Informs J Comput, 2013, 25: 346–363
12 Marinakis Y, Marinaki M. A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Comput Oper Res, 2010, 37: 432–442
13 Toth P, Vigo D. The Vehicle Routing Problem. Philadelphia: SIAM, 2002. 50–82

14 Jiang H, Hu Y, Li Q, et al. Fat computational complexity and heuristic design for the TSP. J Softw, 2009, 20: 2344–2351 [江贺, 胡燕, 李强, 等. TSP 问题的脂肪计算复杂性与启发式算法设计. 软件学报, 2009, 20: 2344–2351]
15 Güner A R, Murat A, Chinnam R B. Dynamic routing under recurrent and non-recurrent congestion using real-time its information. Comput Oper Res, 2012, 39: 358–373
16 Li S C, Liu G. An improved heuristics algorithm for solving traveling salesman problem. J Ind Eng Manage, 2005, 19: 4–8 [李随成, 刘广. 一种改进的 TSP 问题启发式算法. 管理工程学报, 2005, 19: 4–8]
17 Rao W Z, Jin C, Huang Y Y. Two directions moving with difference algorithms to solve traveling salesman problems. J Ind Eng Manage, 2011, 25: 95–102 [饶卫振, 金淳, 黄英艺. 基于求解 TSP 问题的双向扩展差额算法. 管理工程学报, 2011, 25: 95–102]
18 Hassin R, Keina A. Greedy heuristics with regret, with application to the cheapest insertion algorithm for the TSP. Oper Res Lett, 2008, 36: 243–246
19 Held M, Karp R M. The traveling salesman problem and minimum spanning trees. Oper Res, 1970, 18: 1138–1162
20 Helsgaun K. An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur J Oper Res, 2000, 126: 106–130
21 Bentley J J. Fast algorithm for geometric traveling salesman problem. ORSA J Comput, 1992, 4: 387–411
22 Vince A. A framework for the greedy algorithm. Discrete Appl Math, 2002, 121: 247–260
23 Held M, Karp R M. The traveling salesman problem and minimum spanning trees: Part Ⅱ. Math Program, 1971, 1: 6–25
24 Johnson D S, McGeoch L A. The Traveling Salesman Problem and Its Variations: Experimental Analysis of Heuristics for the STSP. Cambridge: MIT Press, 2002. 369–443
25 Pillac V, Gendreau M, Gu é ret C, et al. A review of dynamic vehicle routing problems. Eur J Oper Res, 2013, 225: 1–11
26 Rao W Z, Jin C. An efficient greedy heuristic for solving large-scale capacitated vehicle routing problem. J Ind Eng Manage, 2014, 28: 45–54 [饶卫振, 金淳. 求解大规模 CVRP 问题的快速贪婪算法. 管理工程学报, 2014, 28: 45–54]

附录 A

定理 2 的证明过程

设有 n 个城市的对称 TSP 问题改造后的距离矩阵元素的方差可由式 (A1)∼(A3) 计算.

参数说明:

σ^2 为距离矩阵 D 中元素 (对角线上的元素除外) 的方差;

σ'^2 为改造后的距离矩阵 D' 元素 (对角线上的元素除外) 的方差;

$\bar{d}$ 为距离矩阵 D 中所有元素 (对角线上的元素除外)d_{ij} 的平均值;

d'_{ij} 为改造后距离矩阵 D' 中的元素;

$\bar{d}'$ 为改造后距离矩阵 D' 中所有元素 (对角线上的元素除外) 的平均值;

$sumrow_k$ 为距离矩阵 D 中第 $k(1 \leqslant k \leqslant n)$ 行元素 (对角线上的元素除外) 之和;

$\overline{sumrow}$ 为 n 个 $sumrow_k$ 的平均值;

N 为包含 n 个点的全图中边的数量, 等于 $n \times n$ 矩阵中下 (上) 三角矩阵元素的数量 (对角线元素除外).

$$\sigma'^2 = \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d'_{ij} - \bar{d}')^2}{N}, \tag{A1}$$

$$d'_{ij} = d_{ij} - \pi_i - \pi_j, \tag{A2}$$

$$N = \frac{n(n-1)}{2}, \tag{A3}$$

并且, 有式 (A4) 和 (A5) 成立.

$$\begin{aligned}\bar{d}' &= \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1} d'_{ij}}{N} = \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij} - \pi_i - \pi_j)}{N} \\ &= \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1} d_{ij}}{N} - \frac{(n-1)sum\boldsymbol{\pi}}{N} = \bar{d} - \frac{2}{n}sum\boldsymbol{\pi},\end{aligned} \tag{A4}$$

$$sum\boldsymbol{\pi} = \sum_{i=1}^{n} \pi_i. \tag{A5}$$

因此, 式 (A1) 可以写成式 (A6).

$$\sigma'^2 = f(\pi_1, \pi_2, \ldots, \pi_n) = \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij} - \bar{d} - \pi_i - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})^2}{N}, \tag{A6}$$

在此将方差 σ'^2 看作 n 个变量 $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ 的函数 f.

那么要证明定理 2 相当于证明该连续可导的函数 f 存在全局极小值. 据多元函数求极值法则, 首先应求出 n 个变量的偏导数并令其为零得到有 n 个方程的方程组, 通过解方程组求出判别点; 再检验判别点的 Hessian 矩阵的正定性以确定该判别点是否为极小值.

设 π_k 是其中一个变量 $(1 \leqslant k \leqslant n)$, 为更直观地对变量 π_k 求偏导, 可将函数 f 改写为式 (A7).

$$\begin{aligned} f(\pi_1, \pi_2, \ldots, \pi_n) = \sigma'^2 &= \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d'_{ij} - \bar{d}')^2}{N} = \frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij} - \bar{d} - \pi_i - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})^2}{N} \\ &= \frac{\sum_{i=2}^{n,i\neq k}\sum_{j=1}^{i-1,j\neq k}(d_{ij} - \bar{d} - \pi_i - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})^2 + \sum_{j=1}^{n,j\neq k}(d_{kj} - \bar{d} - \pi_k - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})^2}{N}. \end{aligned} \tag{A7}$$

由式 (A7) 对变量 π_k 求偏导数可由计算式 (A8) 求得.

$$\begin{aligned} \frac{\partial f}{\partial \pi_k} &= \frac{2\sum_{i=2}^{n,i\neq k}\sum_{j=1}^{i-1,j\neq k}(d_{ij} - \bar{d} - \pi_i - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})(\frac{2}{n})}{N} \\ &\quad + \frac{2\sum_{j=1}^{n,j\neq k}(d_{kj} - \bar{d} - \pi_k - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})(-1 + \frac{2}{n})}{N} \\ &= \frac{\frac{4}{n}\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij} - \bar{d} - \pi_i - \pi_j + \frac{2}{n}sum\boldsymbol{\pi}) - 2\sum_{j=1}^{n,j\neq k}(d_{kj} - \bar{d} - \pi_k - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})}{N} \\ &= \frac{\frac{4}{n}\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d'_{ij} - \bar{d}') - 2\sum_{j=1}^{n,j\neq k}(d_{kj} - \bar{d} - \pi_k - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})}{N} \\ &= \frac{\frac{4}{n} \times 0 - 2\sum_{j=1}^{n,j\neq k}(d_{kj} - \bar{d} - \pi_k - \pi_j + \frac{2}{n}sum\boldsymbol{\pi})}{N} \\ &= -\frac{2}{N}\left[\left(\sum_{j=1}^{n,j\neq k} d_{kj}\right) - (n-1)\bar{d} - (n-1)\pi_k - (sum\boldsymbol{\pi} - \pi_k) + \frac{2(n-1)}{n}sum\boldsymbol{\pi}\right] \\ &= -\frac{2}{N}\left[\left(\sum_{j=1}^{n,j\neq k} d_{kj}\right) - (n-1)\bar{d} - (n-2)\pi_k + \frac{n-2}{n}sum\boldsymbol{\pi}\right]. \end{aligned} \tag{A8}$$

令计算式 (A8) 等于零可以解得等式 (A9).

$$\pi_k = \frac{sumrow_k - (n-1)\bar{d}}{n-2} + \frac{sum\boldsymbol{\pi}}{n}, \tag{A9}$$

其中,

$$sumrow_k = \sum_{j=1}^{n,j\neq k} d_{kj}. \tag{A10}$$

由式 (A9) 可得求解判别点的方程组 (A11).

$$\begin{cases} \pi_1 - \dfrac{sum\boldsymbol{\pi}}{n} = \dfrac{sumrow_1 - (n-1)\bar{d}}{n-2} = \dfrac{sumrow_1 - \overline{sumrow}}{n-2}, \\ \pi_2 - \dfrac{sum\boldsymbol{\pi}}{n} = \dfrac{sumrow_2 - (n-1)\bar{d}}{n-2} = \dfrac{sumrow_2 - \overline{sumrow}}{n-2}, \\ \cdots\cdots \quad \cdots\cdots \quad \cdots\cdots \\ \pi_{n-1} - \dfrac{sum\boldsymbol{\pi}}{n} = \dfrac{sumrow_{n-1} - (n-1)\bar{d}}{n-2} = \dfrac{sumrow_{n-1} - \overline{sumrow}}{n-2}, \\ \pi_n - \dfrac{sum\boldsymbol{\pi}}{n} = \dfrac{sumrow_n - (n-1)\bar{d}}{n-2} = \dfrac{sumrow_n - \overline{sumrow}}{n-2}. \end{cases} \tag{A11}$$

设方程组 (A11) 的矩阵表示形式为 $B\pi^{\mathrm{T}}=b$, 则经初等行变换可将增广矩阵 (B | b) 化为最简型矩阵, 变换过程如式 (A12):

$$\left(\begin{array}{ccccc|c} 1-\frac{1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} & -\frac{1}{n} & b_1 \\ -\frac{1}{n} & 1-\frac{1}{n} & \cdots & -\frac{1}{n} & -\frac{1}{n} & b_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -\frac{1}{n} & -\frac{1}{n} & \cdots & 1-\frac{1}{n} & -\frac{1}{n} & b_{n-1} \\ -\frac{1}{n} & -\frac{1}{n} & \cdots & -\frac{1}{n} & 1-\frac{1}{n} & b_n \end{array}\right) \begin{array}{l} r_1-r_n \\ r_2-r_n \\ \longrightarrow \\ r_{n-1}-r_n \\ r_n+\frac{1}{n}(r_1+\cdots+r_{n-1}) \end{array} \left(\begin{array}{ccccc|c} 1 & 0 & \cdots & 0 & -1 & b_1-b_n \\ 0 & 1 & \cdots & 0 & -1 & b_2-b_n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & -1 & b_{n-1}-b_n \\ 0 & 0 & \cdots & 0 & 0 & \frac{1}{n}\sum_{k=1}^{n} b_k \end{array}\right) \tag{A12}$$

其中, $b_k=\frac{sumrow_k-(n-1)\bar{d}}{n-2}(1\leqslant k\leqslant n)$.

注意到在对称距离矩阵中有

$$\begin{aligned} \frac{1}{n}\sum_{k=1}^{n} b_k &= \frac{1}{n}\sum_{k=1}^{n}\frac{sumrow_k-(n-1)\bar{d}}{n-2} = \frac{1}{n}\left[\frac{\sum_{k=1}^{n}\sum_{j=1}^{n,j\neq k} d_{kj}}{n-2}-\frac{n(n-1)}{n-2}\bar{d}\right] \\ &= \frac{1}{n}\left[\frac{\sum_{k=1}^{n}\sum_{j=1}^{n,j\neq k} d_{kj}}{n-2}-\frac{n(n-1)}{n-2}\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1} d_{ij}}{N}\right] \\ &= \frac{1}{n}\left[\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1} d_{ij}}{n-2}-\frac{n(n-1)}{n-2}\frac{2\sum_{i=2}^{n}\sum_{j=1}^{i-1} d_{ij}}{n(n-1)}\right] = \frac{1}{n}\times 0 = 0. \end{aligned} \tag{A13}$$

因此可知方程组 (A11) 的系数矩阵和增广矩阵的秩 R(B)= R(B | b)=$n-1<n$, 方程组 (A11) 有无穷多个解, 且通解如下所示:

$$\boldsymbol{\pi}^c=(\pi_1^c,\pi_2^c,\ldots,\pi_{n-1}^c,\pi_n^c)=c(1,1,\ldots,1,1)+(b_1-b_n,b_2-b_n,\ldots,b_{n-1}-b_n,0). \tag{A14}$$

显然, 方程组 (A11) 的所有解均为函数 f 的判别点, 设 $\boldsymbol{\pi}^c$ 为通解 (A14) 中的任意一个解, 不失一般性, 判别点 $\boldsymbol{\pi}^c$ 对应的 Hessian 矩阵

$$H_f(\boldsymbol{\pi}^c)=\left(\begin{array}{cccc} \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial(\pi_1^c)^2} & \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_1^c\partial\pi_2^c} & \cdots & \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_1^c\partial\pi_n^c} \\ \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_2^c\partial\pi_1^c} & \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial(\pi_2^c)^2} & \cdots & \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_2^c\partial\pi_n^c} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_n^c\partial\pi_1^c} & \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_n^c\partial\pi_2^c} & \cdots & \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial(\pi_n^c)^2} \end{array}\right), \tag{A15}$$

其中, 式 (A15) 主对角线上元素值可由计算式 (A16) 求得.

$$\begin{aligned} \frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial(\pi_k^c)^2} &= \frac{\partial\left(\frac{\partial f(\boldsymbol{\pi}^c)}{\partial\pi_k^c}\right)}{\partial\pi_k^c} = \frac{\partial\left(-\frac{2}{N}\left[sumrow_k-(n-1)\bar{d}-(n-2)\pi_k^c+\frac{n-2}{n}n(c-b_n)\right]\right)}{\partial\pi_k^c} \\ &= \frac{\partial\left(-\frac{2}{N}\left[-(n-2)\pi_k^c+sumrow_k-sumrow_n+(n-2)c\right]\right)}{\partial\pi_k^c} \\ &= -\frac{2}{N}(-(n-2)) = \frac{4(n-2)}{n^2-n}, (1\leqslant k\leqslant n). \end{aligned} \tag{A16}$$

式 (A15) 中非主对角线上的元素值, 如下所示:

$$\frac{\partial^2 f(\boldsymbol{\pi}^c)}{\partial\pi_k^c\partial\pi_h^c} = \frac{\partial\left(\frac{\partial f(\boldsymbol{\pi}^c)}{\partial\pi_k^c}\right)}{\partial\pi_h^c} = \frac{\partial\left(-\frac{2}{N}\left[-(n-2)\pi_k^c+sumrow_k-sumrow_n+(n-2)c\right]\right)}{\partial\pi_h^c} = 0, (h\neq k). \tag{A17}$$

由此求得 $\boldsymbol{\pi}^c$ 的 Hessian 矩阵, 如下:

$$H_f(\boldsymbol{\pi}^c)=\begin{pmatrix}\frac{4(n-2)}{n^2-n} & 0 & \dots & 0\\ 0 & \frac{4(n-2)}{n^2-n} & \dots & 0\\ \dots & \dots & \dots & \dots\\ 0 & 0 & \dots & \frac{4(n-2)}{n^2-n}\end{pmatrix}. \tag{A18}$$

因为在任意 TSP 问题中的城市数量 $n\geqslant 3$, 所以, 对称矩阵 (A18) 的各主子行列式均大于零, 即无论 c 取何值, $H_f(\boldsymbol{\pi}^c)$ 均为正定矩阵. 即式 (A14) 中任意解均能使函数 f 取到局部极小值, 将式 (A14) 中的解代入目标函数 (A6) 计算得到

$$\begin{aligned}f(\boldsymbol{\pi})_{\min} &= f(\boldsymbol{\pi}^c)=\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij}-\bar{d}-\pi_i^c-\pi_j^c+\frac{2}{n}\sum\boldsymbol{\pi}^c)^2}{N}\\ &=\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij}-\bar{d}-(c+b_i-b_n)-(c+b_j-b_n)+2(c-b_n))^2}{N}\\ &=\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij}-\bar{d}-b_i-b_j)^2}{N}\\ &=\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij}-\bar{d}-\frac{sumrow_i-(n-1)\bar{d}}{n-2}-\frac{sumrow_j-(n-1)\bar{d}}{n-2})^2}{N}\\ &=\frac{\sum_{i=2}^{n}\sum_{j=1}^{i-1}(d_{ij}+\frac{n\bar{d}}{n-2}-\frac{sumrow_i+sumrow_j}{n-2})^2}{N}.\end{aligned} \tag{A19}$$

通过式 (A19) 发现, 任何 $\boldsymbol{\pi}^c$ 局部极小值点使目标函数的取值与 c 无关, 即所有 $\boldsymbol{\pi}^c$ 使目标函数的取值均相等. 因此, 求解的任何局部极小值 $\boldsymbol{\pi}^c$ 均为全局极小值。

故定理 2 成立. 证毕.

On the universal strategy for improving a certain type of construction heuristic for the traveling salesman problem

RAO WeiZhen[1*], WANG XinHua[1], JIN Chun[2] & LIU Feng[3]

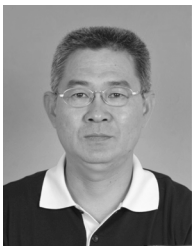
1 *College of Economics and Management, Shandong University of Science and Technology, Qingdao* 266590*, China*;
2 *Faculty of Management and Economics, Dalian University of Technology, Dalian* 116023*, China*;
3 *School of Management Science and Engineering, Dongbei University of Finance and Economics, Dalian* 116025*, China*
*E-mail: raoweizhen@163.com

Abstract In this paper, we analyze the characteristics of four construction heuristics for solving the traveling salesman problem (TSP) and determine that the greatest common weakness of this type of heuristic lies in the greedy construction of solutions during the initial search period. Based on the Held Karp model, we propose the method of Minimizing Variance of Distance Matrix (MVODM) to transform the distance matrix for TSP problems. This method could reduce the greediness of construction during the initial period and improve the overall quality of the solutions. To evaluate the effectiveness of the method, we solve 54 benchmark instances from TSPLIB by using four greedy construction heuristics with and without MVODM for comparison. The results show that MVODM could greatly enhance the solution quality of four heuristics, and some improved heuristics even outperform many world-leading construction heuristics. Moreover, the efficiency of MVODM is so high that an instance size of 2319 cities was run in only 0.076 s on our computer. The increased computation time caused

by MVODM could largely be omitted.

Keywords combinatorial optimization, traveling salesman problem, construction heuristic, computational methods, universal strategy

RAO WeiZhen was born in 1981 and received his Ph.D. degree from the Faculty of Management and Economics, Dalian University of Technology, Dalian, in 2012. Currently, he is a lecturer at the School of Economics and Management, Shandong University of Science and Technology, Qindao, China. His research interests include Logistics Systems Optimization and algorithms for solving TSP and VRP. Dr. Rao is a member of the Systems Engineering Society of China.



WANG XinHua was born in 1960 and received his Ph.D. degree from China University of Mining and Technology, Beijing, China, in 1994. Currently, he is a professor and Ph.D. advisor at the School of Economics and Management, Shandong University of Science and Technology, Qindao, China. His research interests include system evaluation theory and technology and management systems engineering. Dr. Wang is dean of the School of Economics and Management, Shandong University of Science and Technology, as well as the vice director of the System Engineering committee of the China Coal Association.

JIN Chun was born in 1963 and received his Ph.D. degree in Information and Control Engineering from Nagaoka University of Technology, Japan, in 2000. Currently, he is a professor and Ph.D. supervisor at the Faculty of Management and Economics, Dalian University of Technology, Dalian, China. His research interests focus on modeling, simulation, and optimization of logistics and supply chain systems. He is a member of the Systems Engineering Society of China and the Chinese Association for System Simulation as well as a guest professor at Fukushima University, Japan.

LIU Feng was born in 1985 and received his Ph.D. degree from the Faculty of Management and Economics, Dalian University of Technology, Dalian, in 2014. Currently, he is a lecturer at the School of Management Science and Engineering, Dongbei University of Finance and Economics, China. His research interests include scheduling under uncertainties and intelligent algorithm design.