华 中 科 技 大 学

研究生课程考试答题本

考 生 姓 名 李 研

考 生 学 号 M201973122

系 、 年 级 计算机科学与技术学院 2019 级

班 级 硕 1902 班

考 试 科 目 启发式优化

考 试 日 期 2020-01-06

评　　分

题 号	得 分	题 号	得 分
1		9	
2		1 0	
3		1 1	
4		1 2	
5		1 3	
6		1 4	
7		1 5	
8			

总 分 ：	评 卷 人 ：

注：1.无评卷人签名试卷无效。

2.必须用钢笔或圆珠笔阅卷，使用红色，用铅笔阅卷无效。

一种结合数学模型求解旅行采购员问题的启发式算法

摘　　要

随着各大互联网公司的崛起，尤其是菜鸟网络、京东物流、顺丰快递、美团外卖等，他们的具体业务不仅与路由切实相关，而且也会考虑到采购、配送等方面的支出，通常这二者之和占其企业总成本的很大一部分，这一活动在其业务组织和预算中都至关重要。基于该原因，如今采购、物流仍然是一个热门的研究方向。

近年来，组合优化和现代优化计算方法不断发展，将采购和路由二者结合起来联合评估成为一个新的研究方向，旅行采购员问题（TPP）即属于这一方向。针对路由、运输等优化问题的研究，可以追溯到很久以前。这类问题往往表述简单明确、学术性强、求解困难，具有 NP-Hard 或者 NP-Complete 性质。著名的旅行推销员问题（TSP）和车辆路径问题（VRP）即属于这一类。

TPP 要求采购员必须访问一组供应商节点并从其购买产品，以最低的成本开销满足给定的产品需求。成本由两部分构成：路由成本和采购成本。这不仅是企业运营中经常遇到的问题，而且也在生产调度领域的有广泛的应用。由于 TPP 具有 NP 属性，使用精确算法求得最优解十分困难，本文针对经典 TPP 提出一种结合松弛数学模型和启发式搜索的算法，能够比精确算法和已经提出的启发式算法更加快速求解问题。

文章给出了 TPP 的正式定义并建立完整的混合整数规划模型，求解过程中结合松弛模型与经典的启发式算法，如禁忌搜索、扰动启发式等；最后，通过分析实验结果给出算法的优化方案和对 TPP 研究前景的展望。

关键词：旅行采购员问题（TPP）；组合优化；启发式算法；松弛模型

目　　录

1 TPP 定义及分类

1.1 TPP 的定义

TPP 是一个单车路由、采购问题[1]，其定义如下：对于一个仓库 0，一组要购买的产品 K，以及一组地理上分散的供应商 M。对于每一种产品 $k \in K$ 指定一个需求 d_k，找到一个解决方案，使得能够在供应商的子集 $M_k \subseteq M$ 中以 $p_{ik} > 0, i \in M_k$ 的价格购买产品，购买量为 $z_{ik}, i \in M_k$，最终满足需求。此外，对于每一中产品 $k \in K$ 和每一个供货商 $i \in M_k$，定义其产品供应水平为 $q_{ik} > 0$（即 i 供货商可供应 k 产品的上限为 q_{ik}）。注意，为了保证采购计划的可行性，必须满足条件 $\sum_{i \in M_k} q_{ik} \geq d_k, \forall k \in K$。问题场景是定义在一个完整的有向图 $G = (V, A)$ 上，节点集合 $V := M \cup \{0\}$，边集合 $A := \{(i, j) : i, j \in \mathrm{V}, i \neq j\}$，对于每一条边 $(i, j) \in A$ 路径开销为 c_{ij}。TTP 就是在 G 图上以仓库为起止点，寻找一条路径，访问产品供应商的某个子集并决定在每个供应商节点购买多少产品以满足的需求，使得旅行和采购成本最小化。

将 TPP 按照图 1.1 划分为三个层次的子问题，使问题表述更加清晰：

（1） 第一层：供应商节点的选择。并非需要访问所有的供应商节点，这是 TPP 与 TSP 等传统路由问题的一个关键区别。（试想一种极端情况，假若每个供应商节点仅供应一种独一无二的产品，意味着每个供应商节点都要访问，即子问题 1 的解是节点集合 V 的全集，所以 TSP 可以看作是 TPP 的一种特殊情况。）

（2） 第二层：购买方案的分配。一旦确定要访问的供应商节点集合，购买方案的分配是一个可以快速求解的 P 问题，因为只需要按照“最大节约原则”优先从售价最便宜的节点购买产品，直到满足需求即可。

（3） 第三层：路由方案的构建。与第二层子问题类似，一旦确定了访问节点集合，路由方案的构建就转换为传统的 TSP 问题。

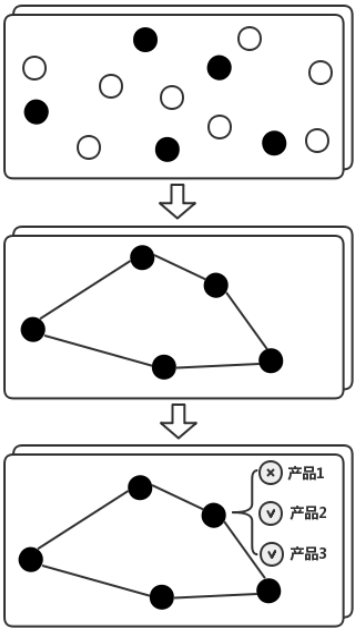

图 1.1　TPP 分层模型

1.2　TPP 的分类

TPP 有两种基本的分类方法[2]。第一种分类是根据 TPP 路由的性质。在有向图中路由成本c_{ij}和c_{ji}可能不同，这种被称为 *Asymmetric TPP*（ATPP）。相反，对于无向图而言，各边满足$c_{ij}=c_{ji}$，该情况被命名为 *Symmetric TPP*（STPP）。可见，STPP 是 ATPP 的一种特殊情况。在相关文献中，ATPP 也通常被称为有向 TPP、非对称 TPP，而 STPP 则被称为无向 TPP 或者对称 TPP。

第二种常见的分类涉及供应商的产品供应水平。如果供应商$i\in M_k$提供产品$k\in K$的数量定义为一个有限值q_{ik}，可能比产品需求d_k小，则 TPP 称为 *Restricted TPP*（R-TPP）。而 *Unrestricted TPP*（U-TPP）则对应供应量是无限的情况，即$q_{ik}\geq d_k,\forall k\in K,\forall i\in M_k$。由此可见，U-TPP 其实是 R-TPP 的一个特例，因为无限制的供给等价于$q_{ik}=d_k=1,\forall k\in K,\forall i\in M_k$。在相关文献中，有的将 R-TPP 和 U-TPP 分别称为 *Capacitated TPP*（C-TPP）和 *Uncapacitated TPP*（U-TPP）。为了避免歧义和混淆，本文将统一采用 R-TPP 和 U-TPP 来描述这种分类情况。

还可以将以上两种分类方法结合起来，从而将 TPP 分为 R-ATPP、R-STPP、U-ATPP、U-STPP 四类。

2 数学模型

2.1 R-ATPP 模型

R-ATPP 模型的决策变量如下：

（1） 设 $y_i, i \in V, V := M \cup \{0\}$ 为布尔变量，当节点 i 被选择时取 1，否则取 0。

（2） 设 $x_{ij}, (i,j) \in A, i \neq j$ 为布尔变量，当有向边 (i,j) 在旅行路径上时取 1，否则取 0。

（3） 设 $z_{ik}, k \in K, i \in M_k$ 代表在 i 供应商节点购买的 k 产品数量。

R-ATPP 模型的目标函数和约束如下：

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{k\in K}\sum_{i\in M_k} p_{ik}z_{ik} \tag{0.1}$$

$$s.t. \quad \sum_{i\in M_k} z_{ik} = d_k, \quad k \in K \tag{0.2}$$

$$0 \le z_{ik} \le q_{ik}y_i, \quad k \in K, i \in M_k \tag{0.3}$$

$$\sum_{j=0}^{n} x_{ij} = \sum_{j=0}^{n} x_{ji} = y_i, \quad i \in V \tag{0.4}$$

$$\sum_{(i,j)\in S} x_{ij} \le |S| - 1, \quad 2 \le |S| < |M_k|, S \subset M_k \cup \{0\} \tag{0.5}$$

$$y_0 = 1 \tag{0.6}$$

$$y_i \in \{0,1\}, \quad i \in M \tag{0.7}$$

$$x_{ij} \in \{0,1\}, \quad (i,j) \in A \tag{0.8}$$

- 目标函数(2.1)的目标是将旅行成本和购买成本的和最小化。
- 约束条件(2.2)保证每种产品的需求都得到准确的满足。
- 约束条件(2.3)要求每个供应商必须访问，即 $y_i = 1$ 时才能从其购买产品，购买数量应为非负值，且不应超过相应的供应量上限。
- 约束条件(2.4)规定了旅行路径的可行性：对于每一个被访问的供应商节点 i，其出度应与入度相等，如果 i 被选中则为 1，否则为 0。
- 约束条件(2.5)是 TSP 子回路约束，通过强制要求在选中节点集合 $M_k \cup \{0\}$ 的任何一个真子集 S 中不形成回路，从而阻止创建子回路。该条约束通常是惰性的，因为子

回路数目庞大，在计算开始时，我们无法将所有的子回路约束都加入到模型中，所以通常是在计算过程中，动态地添加此约束。

- 约束(2.6)要求仓库 0 必须在回路中。
- 最后，约束(2.7)和(2.8)对变量施加布尔约束。

2.2 松弛方法

当一个组合优化问题被判定为 NP-Complete 或 NP-Hard 问题时，解决这个问题常用的方法是构造启发式算法，求出尽量接近最优解的可行解。然而，仅仅有启发式算法是不够的，以极小优化目标函数为例，启发式算法仅仅给出最优值的上界，而松弛方法即是求解下界的一种办法。结合上界与下界，可以提高算法求解效率，同时上界和下界的差值也提供了一种评价目标值的办法。常见的松弛算法有线性规划松弛、拉格朗日松弛等方法。

结合 R-ATPP 模型，我们在求解的初始阶段，无法一次将所有的子回路约束都添加到模型中，只有当求得一个松弛解后检验其可行性，若发现其中包含子回路，则根据式(2.5)添加一条相应的子回路约束。显然，松弛子回路约束扩大了 R-ATPP 可行解区域，随着添加的子回路约束越来越多，可行解区域也逐渐缩小并趋于精确。

因此，在第 2.1 节所构造的模型并非完整模型，只有当所有的子回路约束都被添加到模型中时才构成完整模型。在这之前，求解过程中得的都是松弛解（虽然其中可能已经不包含子回路了，但我们无法证明其具有全局最优性）。

3 一种结合模型的启发式算法

3.1 构造阶段

Step 1. 利用数学求解器（Gurobi）建立求解第 2.1 节的松弛模型，求得松弛解 S。设初始解 $initSln := \{Opt = 0, Tour = \varnothing, PlanTable = \varnothing\}$。

Step 2. *S.Tour* 中可能包含子回路，使用 LKH 算法[5][6]修复回路获得可行解，更新 *initSln*；同时，将子回路约束添加到模型中。

Step 3. 重复 *Step 2*。如果所有的子回路约束都已添加（即完整模型），此时求解器求得的即为全局最优解（$initSln \leftarrow S$）；否则，保留在求解时间区间中获得的历史最优解（$initSln \leftarrow hisBestSln$）。

具体过程如图 3.1 所示：

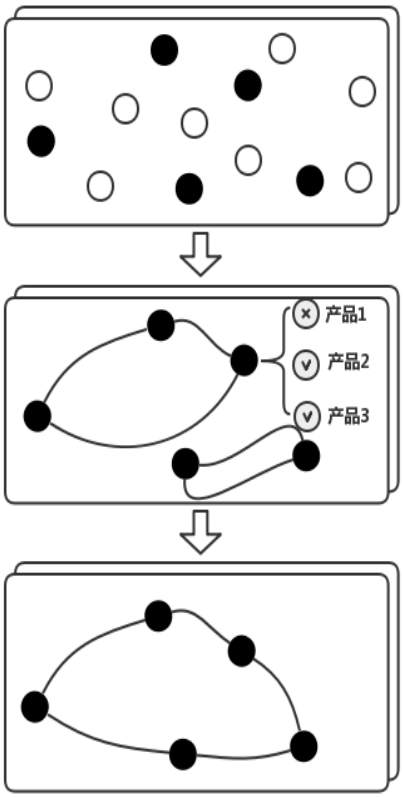

图 3.1 基于模型求解初始解

表 3.1 展示了基于模型求解初始解的构造过程：

表 3.1　构造阶段

Construction Phase	
STEP 1:	$initSln := \{Opt = 0, Tour = \varnothing, PlanTable = \varnothing\}$

STEP 2:	**repeat**
	$S \leftarrow GurobiSolver(Model)$
	if *there are subTour in S.Tour*
	$initSln.Tour \leftarrow LKHRepair(S.Tour)$
	$initSln.PlanTable \leftarrow S.PlanTable$
	$initSln.Opt \leftarrow Update(initSln.Tour,\ initsln.PlanTable)$
	$Model \leftarrow addLazyConstraint(subTour)$
	else
	$initSln \leftarrow S$
	break
	endif
	until no improvement is possible
	return *initSln*

3.2　改进阶段

首先定义如下两个邻域动作：

- Market Drop：依次考虑每个访问过的供应商节点。如果从回路中删除该节点可以降低总成本，那么就删除它。
- Market Add：依次考虑每个未访问过的供应商节点 i。将 i 插入到当前解的回路中，将符合 $p_{jk} > p_{ik}, j \in M_k$ 的产品分配到 i，以降低采购成本。检查新的解决方案，从中删除所有未购买任何产品的供应商节点，保留最终的路由。

改进算法的迭代过程是从一个可行解到另一个可行解变换，通过两个解的比较而选择最好的解，进而作为新的起点进行新的迭代，直到满足一定的要求为止。可以看出，以上两个邻域动作都涉及到路由开销和购买开销之间的权衡，虽然二者的出发点截然相反，但它们的最终目的都是使总开销最小化，所以在一次迭代中要并行评价两个动作生成的所有邻域解，找到最大下降的解前进一步。

在迭代过程中，使用禁忌搜索策略[7]。对于已经探测过的节点，将其在接下来的一段周期中禁忌，使得被删除的节点不会马上添加到回路中，新添加的节点也不会被立刻删除，从而跳出局部最优解。本文设置的禁忌步长如下：

$$\alpha = iter + \frac{|M_k|}{10} + rand\%10$$

$$\beta = iter + \frac{|M| - |M_k|}{10} + rand\%100$$

当一个节点被新添加到回路中时，设其禁忌步长为α；若一个节点被从回路中删除，则设其禁忌步长为β。

注意，由于评价路由部分时采用的是贪心插入/删除的方法，当图上节点关系不满足三角形不等式时，贪心算法并不能保证变更结果是最佳回路，因此最后需要调用 LKH 算法求得精确的路由。同时，考虑到当图的维度较大时，邻域搜索空间也会很大，每次调用 LKH 算法必然会造成较大的时空开销。为了解决该问题，我们采取了以下两种办法：1）引入缓存减少程序的计算量；2）仅对近似评估结果的前 10%调用 LKH 算法修复回路，选择最好的解前进一步。改进阶段如图 3.2 所示：

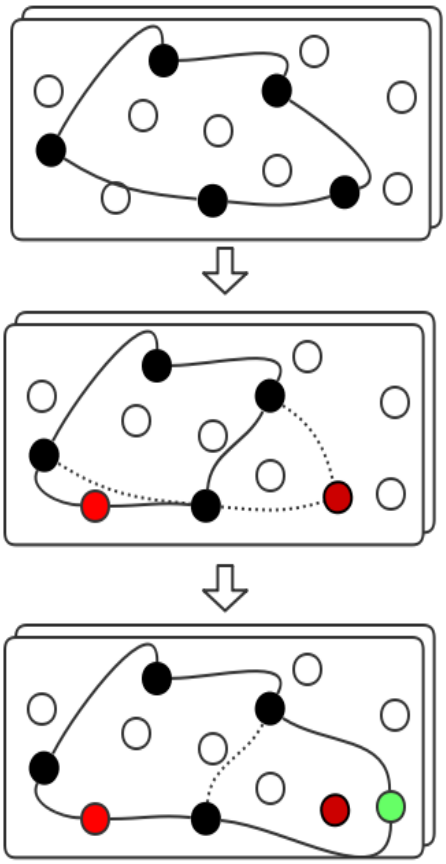

图 3.2　改进阶段

表 3.2 是改进阶段的具体过程：

表 3.2　改进阶段

Improvement Phase	
1:	$s \leftarrow initSln;\ N(s) \leftarrow \varnothing;\ TabuList \leftarrow \varnothing$
2:	**repeat**
3:	$N(s) \leftarrow MarketDrop()$
4:	$N(s) \leftarrow MarketAdd()$
5:	$s \leftarrow TSPHeuristic\left(Top\ 10\ Percent\ Of\ N(s) \setminus TabuList\right)$
6:	$makeMove(s)$
6:	$Update(TabuList)$
7:	**until** stop condition met

4 实验结果

本文的算例来源于 Laporte 等人 2003 年发表的关于 TPP 的精确算法论文《A Branch-and-Cut Algorithm for the Undirected Traveling Purchaser Problem》[4]，文中列举了 4 类 STPP 算例，可以在 http://webpages.ull.es/users/jjsalaza 网站上找到。该网站也收录了大量 ATPP 算例[3]，这些算例已经成为研究 TPP 的经典算例标准集。

200 个节点规模的算例求解结果如表 4.1 所示：

表 4.1 $|V|=200$ 的平均（5 个算例）结果

$\|K\|$	λ	*GAP*(%)	*Total sec*	*Ref sec*
50	0.1	0	889.5	2090.8
50	0.5	0	1041.4	4804.2
50	0.7	0	2391.1	5898.6
50	0.8	0.01	3268.7	4032.4
50	0.9	0.09	1785.6	2164.4
50	0.95	0.2	995.3	14817.2
50	0.99	0.42	735	6253
100	0.1	0	886.2	3460.8
100	0.5	0	905.5	745.75
100	0.7	0	1269.5	1578.667
100	0.8	0	2696.5	4939.4
100	0.9	0.05	2835.4	11925.6
100	0.95	0.14	1314.3	17843.6
100	0.99	0.15	692.7	11872.4
Average		0.08	1550.5	6601.9

100 个节点规模的算例求解结果如表 4.2 所示：

表 4.2 $|V|=100$ 的平均（5 个算例）结果

$\|K\|$	λ	$GAP(\%)$	*Total sec*	*Ref sec*
50	0.1	0	616.7	13.4
50	0.5	0	611.8	16.2
50	0.7	0	647	24
50	0.8	0.02	660.9	92.4
50	0.9	0.06	678.9	409.8
50	0.95	0.1	642.1	786.8
50	0.99	0.03	631.4	98
100	0.1	0	623.7	16.6
100	0.5	0	622.1	17.4
100	0.7	0	640.8	32.8
100	0.8	0.01	626.5	195.4
100	0.9	0.05	640.5	906.4
100	0.95	0.09	637.1	1939.2
100	0.99	0.13	623.4	225.6
150	0.1	0	618.3	19.6
150	0.5	0	628.3	24.8
150	0.7	0	624.3	32.2
150	0.8	0	628.6	41.6
150	0.9	0.02	643.4	1651.8
150	0.95	0.08	657.1	16791
150	0.99	0.08	638.5	688.2
200	0.1	0	619.5	24.4
200	0.5	0	625.6	22.8
200	0.7	0	623.7	24
200	0.8	0	634.8	29.6
200	0.9	0.02	687.6	1315.2
200	0.95	0.06	695.9	10361.4
200	0.99	0.14	676.8	1900.2
Average		0.03	639.5.3	1346.5

50 个节点规模的算例求解结果如表 4.3 所示：

表 4.3 $|V|=50$ 的平均（5 个算例）结果

$\|K\|$	λ	*GAP*(%)	*Total sec*	*Ref sec*
50	0.1	0	4.2	2
50	0.5	0	3.8	2.8
50	0.7	0	3.7	10.6
50	0.8	0	7.1	17.4
50	0.9	0.02	9	50.4
50	0.95	0.01	8.8	16.6
50	0.99	0.02	10.7	10.2
100	0.1	0	3	5.2
100	0.5	0	4	4.8
100	0.7	0	31.9	16.6
100	0.8	0	8.6	11.6
100	0.9	0.01	9.2	46.6
100	0.95	0.03	11.1	38.4
100	0.99	0	7.7	12.4
150	0.1	0	4.3	4.8
150	0.5	0	6.9	5.6
150	0.7	0	5.7	7.4
150	0.8	0	4.4	10.8
150	0.9	0	3.9	66
150	0.95	0.01	5.6	105.6
150	0.99	0	22.7	21.4
200	0.1	0	5.4	6.8
200	0.5	0	6.4	7.2
200	0.7	0	5.6	6
200	0.8	0	8.8	9
200	0.9	0	10.9	80.4
200	0.95	0.01	11.2	220.6
200	0.99	0	12.3	20.8
Average		0.004	8.5	29.2

分析表 4.1 至表 4.3 可以得出以下结论：

（1） 整体而言，结合模型的启发式算法具有更高的求解效率，平均耗时更少，而且算例规模越大，效率优势越明显；

（2） 随着 λ 的增大，求解质量逐渐变差，说明 TPP 的求解难度不仅与算例规模有关，也有产品需求量 d_k 有关。λ 越大，意味着需要在更多的节点购买产品，因此会导致计算量增加。

根据以上分析，我们对应地提出以下改进方案：

（1） 针对 λ 取值较小的情况，目标是继续优化算法求解效率。λ 取值越小，意味着少数节点即可满足购买需求，因此，应该增加调用模型求解的时间，尽量使用模型求得质量较好的解甚至是最优解；同时减少扰动启发式的调用次数，当算法多次迭代仍然不能对当前解有所改进时，应及时终止，同时记录第一次求得最优解的是时间戳。

（2） 针对 λ 取值较大的情况，λ 取值越大，意味着 $|M_k|$ 越大。因此在局部搜索的过程中，每一个供应商节点的添加/删除都会导致购买方案在大量的节点上重新分配，这必然会导致计算量的增加。因此，搜索策略应该适当调整：对于必须访问的节点集合 M^*，拒绝对集合中节点进行任何操作，从而减小需要搜索的邻域空间。

（3） 受 2）的启发，必经点集合 M^* 不仅可以指导搜索策略的改进，同时可以对模型进行完善。将 $y_i = 1, i \in M^*$ 的约束添加到模型中，可以减少模型的运算量，进而提高求解效率。

参 考 文 献

[1] RAMESH T. Traveling purchaser problem[J]. Opsearch, 1981, 18(2): 78－91.

[2] MANERBA D, MANSINI R, RIERA-LEDESMA J. The Traveling Purchaser Problem and its variants[J]. European Journal of Operational Research, 2017, 259(1): 1－18.

[3] SINGH K N, VAN OUDHEUSDEN D L. A branch and bound algorithm for the traveling purchaser problem[J]. European Journal of Operational Research, North-Holland, 1997, 97(3): 571－579.

[4] LAPORTE G, RIERA-LEDESMA J, SALAZAR-GONZÁLEZ J-J. A Branch-and-Cut Algorithm for the Undirected Traveling Purchaser Problem[J]. Operations Research, 2004, 51(6): 940－951.

[5] LIN S, KERNIGHAN B W. An Effective Heuristic Algorithm for the Traveling-Salesman Problem[J]. Operations Research, 1973, 21(2): 498－516.

[6] HELSGAUN K. An effective implementation of the Lin-Kernighan traveling salesman heuristic[J]. European Journal of Operational Research, 1998, 126(81): 106－130.

[7] GENDREAU M, POTVIN J-Y. Tabu Search[G]//Springer, Boston, MA, 2010: 41－59.