巡回セールスマン問題の実用的解法に関する研究*

久保 洋** 沖野 教郎**

Practical Approaches to the Traveling Salesman Problem

Hiroshi KUBO, Norio OKINO

There are many engineering problems founded on the traveling salesman problem, e. g. scheduling problem, routing problem and others. In general, this problem is very difficult to solve practically, because it is one of the combinatorial problem. Applying Branch-and-Bound method, an effective progress for the solution was made by Little and others. These algorithms, however, need too long evaluating time as the number of cities is large, and it is required in practical respect to reduce the computing time. In this paper two new algorithms are proposed which shorten the computing time more than 60 per cent for 20 cities problems, and it is expected that this ratio becomes larger when the number of cities increases. Furthermore, one of these algorithms can be used to an approximate solution of a large number cities problem because its approximation ratio (exact cost/approximate cost) reaches 95 per cent or more in one-tenth in evaluating time compared with the existing methods.

1. 緒 言

工学的な問題の中で、生産加工問題における複数個のドリリング加工の穴あけ順序の決定、交通問題における最適経路の決定など、多くの問題が巡回セールスマン問題として定式化されるか、あるいはこの問題を基礎としている。この巡回セールスマン問題は、最適解を求めることが非常に困難な組合せ最適化問題の一つであり、工業上最適化の程度を進めることが要求されている。すでに種々の解法が提案されており、厳密解を求める解法はほとんどがブランチ・アンド・バウンド法の手法を用い、次の二つに大別される^{1),2)}。

- i) tour building 法 (都市対を逐次選んでゆく)
- ii) subtour elimination 法 (割当て問題を解きながら、逐次サブサイクルを禁止してゆく)

i) は Little³⁾ らによって、ii) は Eastman, Shapiro⁴⁾ らによって開発されたが、いずれの方法も都市数の増加とともに記憶容量より計算時間の方が莫大となり、実用化の面から計算時間の短縮が要求されている。現在のところ計算時間が数分で i) では 40 都市、ii) では 60 都市程度の問題が解かれており、後者の方が優れていると指摘されている。しかしながら、先行関係拘束を含むような実際的な問題では、明らかに前者の tour building の方が有効と思われる⁵⁾。

本論文はこの Little らのブランチングの手法を用い

て、計算時間の短縮および早い時間により良い近似解を得ることが可能な 2 通りの実用的なアルゴリズムを提案するものである。

2. 理論的準備

2.1 巡回セールスマン問題

初めに説明の都合上、巡回セールスマン問題の定式化と次節でその Little らによる解法について述べ、その後で二つの方法で計算時間の短縮などを試みる本論に入る。

巡回セールスマン問題は、 n 個の都市が与えられ、さらにそれら都市間のコストが式 (1) のマトリクス D で与えられたとき、すべての都市をただ一度ずつ通って一周する一巡経路式 (2) の P の中で、式 (3) のコストの総和を最小にするものを求める問題である。

$$(1) D = [d_{ij}] \quad \text{ただし } i, j = 1, 2, \dots, n \text{ かつ } d_{ij} \geq 0$$

$$(2) P = \{(i_1, i_2), (i_2, i_3), \dots, (i_n, i_1)\}$$

$$(3) C = \sum_{(i,j) \in P} d_{ij} \rightarrow \text{最小}$$

ここで d_{ij} は都市 i から j へのコスト、 P の要素である都市対 (i, j) は直接都市 i から j への経路が一巡経路に含まれることを示している。

このほかに、subtour elimination で用いられる割当て問題を用いた定式化もあるが省略する。また一巡経路を求める場合もあるが、簡単に閉路問題に変換できるので、以下では一巡経路は式 (2) のように閉路として扱う。

2.2 Little らによる解法

まずブランチ・アンド・バウンド法の概念を簡単に説明する。これは、ある問題をそれと等価な複数個のより

* 原稿受付 昭和 48 年 11 月 16 日。昭和 48 年度精機学会春季大会学術講演会 (昭和 48 年 4 月 6 日) にて発表。

** 正会員 北海道大学工学部 (札幌市北区北 13 条西 8 丁目)

* S_X に (i, j) を加えると、必ず 1 個のサブサイクルを禁止する都市対が必要となり、これを (k, l) とする。

図 1 ブランチングの状態

簡単な問題に分割すること（ブランチング），およびそれら個々の問題での解以下であることが保証された下界を計算すること（バウンディング）の繰返しから構成される。さらにブランチングしたある問題の下界が可能解であると、それを上界としてその値以上の下界をもつ問題には最適解がないことが保証され、上界以上の下界をもつ問題がなくなるまでにこれらを繰返すことにより最適解が求まる。

Little らによる解法のブランチングの方法について説明する。いま図 1 に示すように、ある問題（以下ノードと呼ぶ） X ですでに選ばれている都市対の集合を S_X ，すでに禁止されているかまたは S_X によるサブサイクルを禁止する都市対の集合を R_X とすると、ノード X ではこの S_X と R_X に矛盾しないすべての一巡経路の中で最適解を求める問題となっている。ブランチングはこのノード X でまだ決められていないある都市対 (i, j) について、これを必ず一巡経路に含むノード Y と、必ず含まないノード $\bar{Y}$ とで行われる。このときノード Y での S_Y ， R_Y とノード $\bar{Y}$ での $S_{\bar{Y}}$ ， $R_{\bar{Y}}$ は、 S_X ， R_X および (i, j) よりそれぞれ図中のようになる。また都市対が $n-2$ 個選択されると、そのノードの下界は必ず可能解となり、このとき上界の決定とノードの消去が行われる。

次に下界の計算法について説明する。いまあるノード X でブランチングを開始するとして、この下界 $w(X)$ は式 (4) で縮退マトリクス D_X を作成したのち、式 (5) で計算される。

$$(4) \quad i) \quad \bar{D}_X = [\bar{d}_{ij}^X] \quad (i, j) \in R_X \text{ のとき } \bar{d}_{ij}^X = \infty,$$

$$\text{その他 } \bar{d}_{ij}^X = d_{ij}$$

$$ii) \quad \hat{D}_X = [\hat{d}_{ij}^X]$$

$$i \in I_X \text{ について } \hat{d}_{ij}^X = \bar{d}_{ij}^X - \min_{p \in J_X} \bar{d}_{ip}^X$$

$$iii) \quad D_X = [d_{ij}^X]$$

$$i \in J_X \text{ について } d_{ij}^X = \hat{d}_{ij}^X - \min_{p \in I_X} \hat{d}_{ip}^X$$

$$\text{ただし } I_X = \{i | (i, j) \in S_X\}, J_X = \{j | (i, j) \in S_X\}$$

$$(5) \quad w(X) = \sum_{(i,j) \in S_X} d_{ij} + \sum_{i \in I_X} \min_{j \in J_X} \bar{d}_{ij}^X + \sum_{j \in J_X} \min_{i \in I_X} \hat{d}_{ij}^X$$

ここで I_X は消去した行， J_X は列の集合であり、この操作によって D_X の消去されていない行と列には、それぞれ 1 個以上のゼロ要素が存在する。 $w(X)$ が上界より小さければ、これらすべてのゼロ要素について式 (6) で選択指標の $\theta(i, j)$ を計算し、この値の最大な都市対 (i, j) についてブランチングが行われる。この都市対を禁止したノード $\bar{Y}$ での下界 $w(\bar{Y})$ は式 (7) で計算され、選択したノード Y では式 (8) で $\bar{D}_Y$ を、さらに式 (4) ii) iii) で D_Y を作成したのち式 (9) で計算される。

$$(6) \quad \theta(i, j) = \min_{p \in J_X \cup j} \bar{d}_{ip}^X + \min_{p \in I_X \cup i} \hat{d}_{pj}^X$$

$$\text{ただし } \bar{d}_{ij}^X = 0 \text{ かつ } i \in I_X, j \in J_X$$

$$(7) \quad w(\bar{Y}) = w(X) + \theta(i, j)$$

$$(8) \quad \bar{D}_Y = [\bar{d}_{ij}^Y] \quad (i, j) \in R_Y \text{ かつ } (i, j) \in R_X \text{ のとき}$$

$$\bar{d}_{ij}^X = \infty \quad \text{その他 } \bar{d}_{ij}^X = d_{ij}^X$$

$$(9) \quad w(Y) = w(X) + \sum_{i \in I_Y} \min_{j \in J_Y} \bar{d}_{ij}^Y + \sum_{j \in J_Y} \min_{i \in I_Y} \hat{d}_{ij}^Y$$

以下この選択したノード Y からすぐ次のブランチングを続ける場合は式 (9) が繰返し使われ、新たなノードからブランチングを開始する場合のみ、式 (5) が下界の計算に使われる。この次のブランチングを行うノードの決め方も種々の方法があるが、主なものは次の二つである。一つは、常に未ブランチングで最小の下界をもつノードから行う方法で、他は可能解が得られるまで選択したノードから続けてブランチングを行う方法である。

2.3 非減算法

前節で明らかなように、従来の方法では下界の計算および選択指標の計算に、非常に多くのマトリクスの減算と最小値の計算を含んでいる。まず計算時間の短縮法として、この節では縮退の際のマトリクスの行および列方向の減算の省略を試み、次節でさらにこれら最小値の計算の省略を試みる。

ブランチングを開始するときのノード X での下界は、式 (4) の $\bar{D}_X$ を使って式 (10) で表され、明らかにマトリクスの減算を省略しても $w(X)$ の計算が可能である。しかし第 3 項に減算を含むので、これを除いた式 (11) で新たな下界 $v(X)$ を定義する。これは常に $w(X)$ 以下となり、明らかに下界の条件を満たしている。

$$(10) \quad w(X) = \sum_{(i,j) \in S_X} d_{ij} + \sum_{i \in I_X} \min_{j \in J_X} \bar{d}_{ij}^X + \sum_{j \in J_X} \min_{i \in I_X} \{ \bar{d}_{ij}^X - \min_{p \in I_X} \bar{d}_{ip}^X \}$$

$$(11) \quad v(X) = \sum_{(i,j) \in S_X} d_{ij} + \sum_{i \in I_X} \min_{j \in J_X} \bar{d}_{ij}^X \leq w(X)$$

式(6)の選択指標 $\theta(i, j)$ をこの $\bar{D}_X$ から計算することは, 多数の減算を含んだ最小値の計算があること, 下界の値自身が変えることから実用上不可能である。したがって下界 $v(X)$ を使うときの選択指標を新たに $\phi(i, j)$ として, 式(12)を満たすすべての (i, j) について式(13)で定義する。

$$(12) \quad \bar{d}_{ij}^X = \min_{p \in J_X} \bar{d}_{ip}^X, \quad \text{ただし } i \in I_X$$

$$(13) \quad \phi(i, j) = \min_{p \in J_X \cup j} \bar{d}_{ip}^X - \bar{d}_{ij}^X$$

これが選択指標の条件を満たすことを示すには, 式(14)の $v(X)$ が (i, j) を禁止したノード $\bar{Y}$ での下界の条件を満たしていることを示せばよい。これは明らかに, 式(11)の $\bar{D}_X$ で $\bar{d}_{ij}^X = \infty$ としたときに一致している。また (i, j) を選択したノード Y での下界は, 式(8)で $\bar{D}_Y$ を作成したのち式(11)で計算される。

$$(14) \quad v(\bar{Y}) = v(X) + \phi(i, j) \leq w(\bar{Y})$$

この新たに定義した下界は, 減算の省略ばかりでなく, 従来行と列両方向でゼロ要素を持つように下界を計算していたものを, 行方向のみの半分にしたことに相当している。したがって, 従来の最近接パス法に下界と分岐の手法を適用した形となっており, 1個の可能解を非常に短い時間で計算可能である。しかしながら, このままでは各ノードでの下界が小さくなり, ブランチングが非常に増加してしまう。これは列方向の大小関係を無視したためであり, このためにブランチングを開始するノード X では式(15)で列方向のみのマトリクスの減算を行っておくことにする。

$$(15) \quad j \in J_X \text{ について } \bar{d}_{ij}^X = \bar{d}_{ij}^X - \min_{p \in I_X} \bar{d}_{ip}^X$$

したがってノード X 以下では各列から減算した値の和を式(11), (14)の下界に常に加える必要があるが, ブランチングを開始するノード X では従来の下界と同じになり, ブランチングの増加を抑えられる。

2.4 インデックス法

従来の方法, 前節での非減算法いずれにおいても, 下界と選択指標の計算に多数の最小値の計算を含んでいる。以下の説明には非減算法を使って, この最小値の計算のできるかぎりの省略を試みる。

いま, あるノード X でブランチングを開始するとして, 式(4) i), (15)で $\bar{D}_X$ を作成したのち式(16)を満たす $2 \times n$ マトリクス E_X を作成する。これらの要素は, その行が消去されていればゼロ, その他はその行の最小の要素と2番目に小さな要素の指標すなわち列の番号で構成

されている。したがって式(13)の $\phi(i, j)$ は式(17)となり, 最小値の計算がなくなる。

$$(16) \quad E_X = [e_{ai}^X] \quad i \in I_X \text{ のとき } e_{1i}^X = e_{2i}^X = 0$$

$$i \in I_X \text{ のとき } e_{1i}^X = p, \quad e_{2i}^X = q$$

$$\text{ただし } p, q \text{ は } \bar{d}_{ip}^X = \min_{j \in J_X} \bar{d}_{ij}^X, \quad \bar{d}_{iq}^X = \min_{j \in J_X \cup p} \bar{d}_{ij}^X$$

$$(17) \quad \phi(i, j) = \bar{d}_{ip}^X - \bar{d}_{ij}^X \quad i \in I_X \text{ について } i = e_{1i}^X$$

さらにこの E_X から, 都市対 (i, j) が選択されたノード Y での E_Y が少ない最小値の計算で作れることを示す。いま $\bar{D}_X$ で消去も禁止もされていない要素 (k, l) が禁止されたとすると, このときの $\bar{E}_X$ は式(18)のアルゴリズムで E_X から修正できる。ここで必要な場合にのみ最小値の計算が行われることに注意されたい。したがって, $\bar{D}_Y$ を作る際に $\bar{D}_X$ で禁止される式(19)を満たすすべての都市対について式(18)を行ったものを E_Y とすれば, $\bar{D}_Y$ から式(16)で作成したものと一致する。ただし $e_{1i}^X = e_{2i}^X = 0$ とする。

$$(18) \quad \text{i) } e_{1k}^X = l \text{ かつ } e_{2k}^X = l \text{ のとき } \bar{E}_X = E_X$$

$$\text{ii) } e_{1k}^X = l \text{ のとき } \bar{e}_{1k}^X = e_{2k}^X \left\{ \begin{array}{l} \bar{e}_{2k}^X = p \\ \bar{e}_{1k}^X = e_{1k}^X \end{array} \right. \quad \bar{e}_{2k}^X = p$$

$$\text{iii) } e_{2k}^X = l \text{ のとき } \bar{e}_{1k}^X = e_{1k}^X \left\{ \begin{array}{l} \bar{e}_{2k}^X = p \\ \bar{e}_{1k}^X = e_{1k}^X \end{array} \right. \quad \bar{e}_{1k}^X = p$$

ただし $\bar{d}_{kp}^X = \min_{q \in J_X \cup e_{1k}^X} \bar{d}_{kq}^X$ 。他の要素はそのまま。

$$(19) \quad \text{i) } (k, l) \in R_Y \text{ かつ } (k, l) \in R_X$$

$$\text{ii) } k \in I_Y \text{ について } (k, j)$$

以下同様に選択のブランチングを続けてゆくときは, 必要な場合のみ最小値を計算するだけで下界と選択指標の計算が可能となる。

従来の方法で下界を計算する場合にも, 同様のインデックステーブルを使って減算の必要な行あるいは列を判定したり, 最小値の計算を必要な場合にのみ行うようにすることができる。この場合は行方向のテーブル E_X のほかに, 同じような列方向のインデックステーブル F_X をも使う必要があり, 式(18)のような修正のアルゴリズムはもっと複雑になる。詳細については省略するが, 最小値の計算は大幅に減少される。筆者らは, この従来の下界で最小値の計算を減少したものをインデックス法と呼び, この節で説明した非減算法に適用したものはこれと区別するためにそのまま非減算法と呼んでいる。これは, 従来の方法, 前節の非減算法いずれの場合でもインデックステーブルを用いた方が明らかに計算時間の短縮が予想されるためである。すなわち, インデックステーブルを用いることは手法的な改良であり, 下界および選択指標の値はいずれも適用前と同じである。

図2 フローチャート

これら非減算法, インデックス法いずれの方法でも最適解が求まることを以下に示しておく. ブランチ・アンド・バウンド法の最適性を示すには, ブランチングの最適性すなわち各ノードが等価な複数個のノードに分割されることと, バウンディングの最適性すなわち各ノードでの下界が最適解以下であることの二つを示せば十分である. ブランチングについてはインデックス法, 非減算法いずれも Little らの方法と同じなので, 明らかに最適性が保証される. バウンディングについては, インデックス法は下界, 選択指標ともに従来の方法での値と同じなので明らかである. 非減算法の下界と選択指標は式(11)と式(14)でそれぞれ示されるように従来の方法での値以下であり, 明らかにバウンディングの最適性も保証される.

3. アルゴリズムと実験

3.1 アルゴリズム

従来の方法および非減算法, インデックス法のフローチャートを図2に示す. プログラムはフォートランを用い, 三つの方法すべてを1個のサブルーチンとして作成し, 二重わくの部分の処理のみを引数を使って変えるようにした. すなわち下界と選択指標の計算部分が, それぞれ前節の説明のように異なるだけである. また図中の破線わくの部分は, 常に下界の一番小さなノードからブランチングを開始するところで, これを除くと Little ら

表1 コストマトリックスの一例 (10都市)

$i \backslash j$	1	2	3	4	5	6	7	8	9	10
1		8	3	4	9	17	16	18	10	6
2	2		8	16	18	19	2	9	10	14
3	3	7		10	10	7	15	19	1	20
4	16	10	14		1	1	11	8	5	17
5	18	15	14	0		4	18	8	3	12
6	15	14	6	18	15		16	7	10	19
7	9	13	10	1	20	4		12	9	8
8	14	20	2	6	7	14	10		19	2
9	5	19	5	9	17	5	12	11		5
10	17	3	4	18	17	13	15	18	20	

が go to right と呼ぶ上界が得られるまで選択のみを続けてブランチングしてゆく方法となる. 筆者らは次の二つの理由で, この破線部を省略する方法 (以下では分岐拘束法と呼ぶ) を推奨する. まず第一に, 都市数の増加ばかりでなくコストマトリックスの性質により計算終了時には大きなばらつきがあり, 分岐を拘束しないと1個の可能解も得られずに計算打ち切りの場合が生ずる. 第二に, インデックス法で明らかのように一度ブランチングを開始してインデックステーブルを作成すると, 分岐を拘束した方が計算時間の短縮となる.

ノードの格納法については, 1個の都市対につきブランチング後常に1個ずつある選択と禁止のノードはまとめて, かつポインタを用いてリスト処理的に記憶し, 記憶容量の節約と計算時間の短縮を図った.

3.2 実験と結果

前節でのプログラムを使用して, 従来の方法と新たな非減算法, インデックス法について主に次の2点について比較するため, 例題を作成し実験を行った.

I 計算終了時間, ノード数などの比較

II 上界の近似度と, その出る時間の早さの比較

I は普通に考えられる比較であるが, II については分岐拘束法を提唱した理由と同じく, いつ計算時間を打ち切ってもできるだけ良い近似解が得られている解法が望ましいという観点から, これを比較するものである. 例題として解くコストマトリックスは, 計算機組込み関数の一様乱数発生ルーチンを使って, 10都市, 20都市問題をそれぞれ10個ずつ作成した. いずれも乱数の平均値を都市数に等しくした. 10都市の一例を表1に示す. 使用した計算機は, 北海道大学大型計算機センターの FACOM 230/60 である.

I の場合の結果として10都市を表2, 20都市を表3に示す. 従来の方法のままでも, 分岐拘束法の方がノー

表2 10都市での結果

		従来の方法		インデックス法		非減算法
分岐拘束の有無		なし	あり	なし	あり	あり
計算終了時間 (s)	最大	0.77	0.73	0.48	0.42	0.64
	平均	0.42	0.40	0.26	0.24	0.20
	最小	0.24	0.24	0.14	0.13	0.06
解のでの時間(s)		0.38	0.31	0.24	0.18	0.12
分岐開始数		3.3	2.4	3.3	2.4	4.9
上界の個数		1.0	1.4	1.0	1.4	1.7
ノード数 (1/2)		13.2	13.4	13.2	13.4	20.9

表3 20都市での結果

		従来の方法		インデックス法		非減算法
分岐拘束の有無		なし	あり	なし	あり	あり
計算終了時間 (s)	最大	17.02	11.52	10.63	5.42	7.49
	平均	9.36	8.73	5.24	3.70	3.38
	最小	4.65	4.48	2.18	2.66	1.10
解のでの時間(s)		8.31	5.24	4.65	2.01	1.92
分岐開始数		31.1	16.3	31.1	16.3	27.8
上界の個数		1.0	2.5	1.0	2.5	3.0
ノード数 (1/2)		82.4	95.5	82.4	95.5	170.0

ノード数の若干の増加はあるが、計算終了時間は短縮される。インデックス法の場合はこの短縮の比がさらに顕著になり、また計算終了時間のばらつき、すなわち最小と最大の間が小さくなる。計算終了時間、解のでの時間いずれも平均時間では非減算法が一番短い、この終了のばらつきがインデックス法より大きい。この理由は、非減算法が行方向のみで下界を計算するための影響と思われる。従来の方法で10都市問題で平均0.4秒、20都市で9.4秒程度かかっていたものが、新たな方法ではそれぞれ0.2秒、3.4秒程度となり、10都市で50%、20都市で65%程度計算終了時間が短縮された。このことは、都市数の増加とともにさらに計算時間が短縮することを予想させる。なお計算時間はCPU timeである。

IIの場合の結果として、20都市の場合を図3に示す。コスト、時間いずれをも規準化して平均するために、コストについては厳密解/上界、時間については従来の分岐拘束のない方法での終了時間を1.0として、出てくる上界の近似度と早さをプロットした。さらに時間の方については0.05刻みで量子化してある。明らかに分岐拘束を行うと、非常に早く近似解として使える上界が得られる。特に非減算法では、終了時間の10%程度の計算時間で近似度が0.96以上となり、近似解法としても使えよう。また最初の可能解の得られる時間を100都市問題で比較すると、従来の方法では約160秒、インデック

図3 近似解の出る早さの比較 (20都市)

ス法では約20秒、非減算法では約5秒程度で、それぞれ近似度は若干悪くなるが非減算法が非常に早い。

4. 結 論

本研究で得られた成果を以下にまとめる。

- (1) 巡回セールスマン問題の実用的な計算法を、新たに二つ提案した。Littleらの方法で、できる限り最小値の計算を減少した方法をインデックス法、マトリクスの減算までを省略し新たな下界を導入した方法を非減算法と名付けた。
- (2) 実験を通して新たな方法の実用性をも示し得た。計算時間は20都市問題で従来の65%程度が短縮され、都市数の増加とともにさらに短縮することが予想される。
- (3) 分岐拘束法の有効性が明らかとなった。従来の方法においても若干のノード数の増加のみで、計算時間は短縮される。
- (4) 非減算法は、都市数が増加した場合などには近似解法として使用可能であろう。20都市問題では、従来の計算終了時間の1割程度で95%以上の近似度に達する。

おわりに、本研究に際し日ごろご検討・ご協力いただきました当精密工学科精密機器学第一講座の町田史彦氏(現第二精工舎)ならびにその他の諸氏に心から感謝いたします。

参 考 文 献

- 1) M. Bellmore and G.L. Nemhauser: The Traveling Salesman Problem; A Survey, Opns. Res., 16 (1968).
- 2) E.L. Lawler and D.E. Wood: Branch-and-Bound Method; A Survey, Opns. Res., 14 (1966).
- 3) J.D.C. Little, K.G. Murty, D.W. Sweeney and C. Karel: An Algorithm for the Traveling Salesman Problem, Opns. Res., 11 (1963).
- 4) D. Shapiro: Algorithms for the Solution of the Optimal Cost Traveling Salesman Problem, Sc. D. Thesis, Washington Univ., St. Louis (1966).
- 5) 町田史彦, 嘉数侑昇, 沖野教郎: 先行関係拘束のある順序付け問題の一解法, 精密機械, 40, 3 (1974) 233.