

交通ネットワークにおける対災害信頼性の最適配分

神戸大学工学部 正員 枝村 俊 郎
神戸大学工学部 正員 森 津 秀 夫
片平エンジニアリング〇正員 土 井 元 治
神戸大学大学院 学生員 中 川 勝八郎

1. はしがき

地震、風水害あるいは積雪というような大規模な自然災害がある地域を襲ったとき、なおかつその地域の交通網が健全に機能し続けることが望ましいことはいうまでもない。しかしながら、限りある予算のもとで交通網を構成するすべての要素をこれらの自然災害に対して最大の信頼性を与えるように設計し維持することはしばしば困難であり、また資源の浪費であるといえよう。

この研究では、一定の予算制約下で交通サービスを災害時においても十分発揮できるように、すなわち災害に対する信頼性を最大に保つように、交通網を構成する各リンクの対災害信頼性の水準を最適に配分する問題をとり扱う。ここにいう対災害信頼性は規格のよりにみなされ、その水準は離散的に与えられるものとする。

電気、機械装置や構造物などのシステムが規定された機能を遂行しうる確率についての研究、いわゆる信頼性理論はすでに相当の成果がみられる。しかしリンクを構成要素とみなす場合の、交通網のような複雑なシステムの信頼性については、これまでほとんど研究がなされていない。

ここでは、まず最初に交通ネットワークの対災害信頼性の概念について述べ、次にこの問題を2つの場合に分けて定式化を行う。ひとつは、ある災害に対してネットワークの破壊パターンが1つに決まると仮定してその総所要時間を評価関数とする場合である。他のひとつは、ある災害に対してネットワークの状態が確率的に決まると仮定し、目的地に到着できるトリップ数の期待値で評価する場合である。そしてそれぞれの最適解を求めるアルゴリズムを提案し、いくつかの計算例を示す。

2. 交通ネットワークの対災害信頼性の概念

(1) 交通ネットワークの対災害信頼性の評価について

交通ネットワークに災害が加わり、その一部のリンクが使用不能になればネットワークの状態は平常時と異なったものとなる。多くのリンクが同時に破壊され使用可能なリンクが減少すれば、当然トリップの所要時間は増大し、ネットワーク全体として大きな不利益を被ることになる。そのほかほほだしきにはネットワークが非連結となり、目的地に到着できないトリップがでてくることも考えられよう。

そこで、われわれは対災害信頼性の高い交通ネットワークとは、平常時のみならず災害時においても〇ロトリップの存在するノード間にパスが存在し、その所要時間が小さいものであると考える。ただしここにいう災害とはリンクに損傷を与え使用不能になさしめる可能性のあるものの総称とする。

災害に対してリンクが破壊するかどうかは、リンクの強度と災害の強度の相対的な関係によって決まってくると考えられる。しかし破壊のメカニズムは非常に複雑なため、リンクの強度や災害の強度を完全に把握することは困難である。このため、現状では災害に対してリンクが破壊するかどうかは確率的に決まるとすることにより、不十分な点を補わざるをえない。このとき、前述の意味で対災害信頼性の最適化を行うならば、ネットワークを利用するトリップの総所要時間の期待値を評価関数とするのが妥当であろう。

リンクが破壊するかどうかは確率的に決まるということは、そのリンクがどちらの状態をもとりうることを意味する。たとえば m 本のリンクから成るネットワークでは、 2^m の破壊パターンがそれぞれの生起確率をもって得ら

れることになる。そして各パターンの総所要時間に生起確率をかけた和として評価関数である期待値が求まる。しかしこれでは1つの解についての評価関数値を求めるのにさえ、計算時間がかかり過ぎるため最適化を行うことはできない。よって簡単のため次の2つの場合に分けて定式化することにする。

ひとつは、ある災害のもとでネットワークのリンクが破壊するかどうかが決まると仮定する。そして次に述べる災害パターンの概念を用い、いくつかのネットワークの状態での総所要時間の重みづけした総和を最小化する問題である。他のひとつは、ある災害のもとで、ネットワークのリンクが破壊するかどうか確率で与えられると仮定する。そしていくつかの災害パターンに対して目的地に到達できるトリップ数の期待値の重みづけした総和を最大化する問題である。

次に定式化の際に導入したいくつかの新しい概念について述べる。

④ リンクレベル、災害強度レベルおよび災害パターンの概念の導入

リンクの対災害信頼性の水準を離散的な規格として表わすリンクレベルを設定する。費用はリンクレベルの関数とし、高いリンクレベルのリンクを建設する際には高い費用がかかるものと考えるのである。このことにより、問題は一定の予算制約下で最適なリンクレベルの組合せを決定する整数計画問題として定式化される。

ある地域に展開している交通ネットワークに自然災害が及ぶとき、近接するリンクには同じような影響を与えるであろう。すなわち災害の強度はリンクに独立ではなく、ある地域的分布をもつものと考えられる。地域的分布は災害ごとに変化するであろうが、これらはいくつかの代表的なパターンに分類することが可能であると仮定する。ここではこのパターンを災害パターンと呼ぶことにする。すなわち各リンクには各パターンに応じた固有の災害強度が加わるものと考えるのである。また各リンクに対する災害の強度は、ここでは簡単のために離散的な災害強度レベルで表わすことにする。

総所要時間最小化問題の場合、あるリンクについて災害強度レベルがリンクレベル以上であれば、そのリンクは破壊するものとする。たとえば図-1のネットワークが災害を受ける前の状態で、そのリンクレベルは()内の数字で表わされているとする。そして[]内の数字で表わされる災害強度レベルをもつ災害パターンが加わったときには、ネットワークは図-2のようになるとする。

また到着トリップ数最大化問題においては、リンクレベルと災害強度レベルの組合せに対してそれぞれリンクの破壊確率あるいは非破壊確率が与えられるものとする。

3. 総所要時間最小化問題

(1) 問題の定式化

各災害パターンによりネットワークの状態は変化する。よってこれらの総所要時間の重みづけした和を評価関数とし、建設費制約の下で最適化を行う。

$$\min Z_1 = \sum_{j=1}^m (w_j \sum_{i=1}^n g_{ij} \cdot t_{ij}(x_i, \phi_j)) \quad (1)$$

$$s.t. \quad \sum_{k=1}^m C_k(x_k) \leq C^u \quad (2)$$

$$L_k \leq x_k \leq L_k^{max} \quad (k=1, 2, \dots, m) \quad (3)$$

ここに、 Z_1 : 目的関数で、災害パターンにより重みづけした総所要時間の和。 t_{ij} : ノード i からノード j への最短所要時間。ただしノード i からノード j へのパスが存在しないときは、所要時間のかわりにペナルティーを与える。 w_j : 災害パターン j の重み。 $\sum_{j=1}^m w_j = 1$ 。 $x = x = (x_1, x_2, \dots, x_k, \dots, x_m)$ 。検討リンクのリンクレベルを表わす変数。正整数である。 $x_k = 0$ はリンク k が存在しないことで

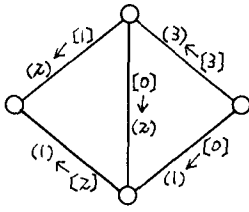


図-1 災害を受ける前のネットワーク

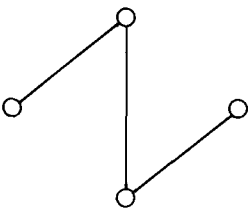


図-2 災害を受けた後のネットワーク

$L_k = L^{max}$ はリンク k が最高のリンクレベルにあることを表す。いまかりに、 $L_k = 2$ とすると、リンク k は災害強度レベル m 以上の災害によって使用不能となる。 $\phi_s: \phi_s = (\phi_{s1}, \phi_{s2}, \dots, \phi_{sm}, \dots, \phi_{ms})$ 。災害パターン s の各リンクに対する災害強度レベルを表す正整数である。 $\phi_{ks} = 0$ は災害パターン s のリンク k に対する災害がないことを表す。よって平常時の災害パターンは、 $\phi_{ks} = 0$ ($k = 1, 2, \dots, m$) となる。 $C_k(L_k)$: リンク k のリンクレベルを L_k から L_k に改善するのに必要な費用。 C^u : 建設費の上限値。 s : 災害パターン数。 L_k : リンク k の改良する前のリンクレベル。 L^{max} : リンクを改良できる最高のリンクレベルとする。

式(1)において、 $\delta_{ij} \cdot t_{ij}(x, \phi_s)$ はリンクレベルが x 、災害パターンが ϕ_s のときの、ノード i からノード j への最短所要時間にトリップ数をかけたものである。それをすべてのノード対について合計し、災害パターン s の下での総所要時間を求める。そしてすべての災害パターンに対して総所要時間の加重和をとったものが式(2)である。式(2)はリンクを改善する際の全費用が一定の予算 C^u を上まわらないための制約である。式(3)はリンクレベルが現在のリンクレベル以上であり、かつ考えている最高のリンクレベル以下であることを示す。

(2) 目的関数の下限値について

この問題は目的関数に総所要時間を使用した整数計画問題であるという点で従来のいわゆる最適ネットワーク問題と一致している。もし災害パターンとして平常時のみを考え、その重みを1にし、リンクレベルを0か1とすれば、まったく同じ問題になる。

一般にこれらの整数計画問題は、可成り大きなネットワークになると実行可能解の数が膨大になり計算が不可能になるという特徴がある。そこで、Hoang¹⁾ は目的関数の下限値を使用することにより無駄な目的関数の計算を省略するアルゴリズムを提案している。さらに松村・森津²⁾ は下限値を求めるときのLP解からいっせいに実行可能解を得ることにより、組合せトリー上でより早い段階に最適解を求めるアルゴリズムを開発している。式(1)~(3)の問題は複数のリンクレベルで表わされている点で従来の最適ネットワーク問題とは異なっている。このことは解がリンクレベルの組合せであることから実行可能解の増加にもつながる。また目的関数の計算に際しても各災害パターン F でのネットワークに対して最短路探索問題を解かなければならず、これも計算時間の増大をもたらす。このため目的関数の下限値を求め計算の効率化をはかることはより重要である。本研究であつた問題は複数のリンクレベルが存在するので、Hoang の下限値をそのまま適用することはできない。そこで、簡単に目的関数の下限値と他のLP解を求められるよう、あらたに変換操作アルゴリズムを開発する。次に述べる補助問題は目的関数の増加の下限値を求める定式化を行うものである。

補助問題1

$$\min F = \sum_{s=1}^m \sum_{k=1}^{L^{max}-L_k} f_{kl} \cdot y_{kl} \quad (4)$$

$$\text{s.t.} \quad \sum_{k=1}^m \sum_{l=1}^{L^{max}-L_k} C_{kl} \cdot y_{kl} \geq C^{max} - C^u \quad (5)$$

$$y_{kl} \geq y_{k, l+1} \quad \left(\begin{array}{l} k=1, 2, \dots, m \\ l=1, 2, \dots, L^{max}-L_k \end{array} \right) \quad (6)$$

$$y_{kl} = 0 \text{ or } 1 \quad \left(\begin{array}{l} k=1, 2, \dots, m \\ l=1, 2, \dots, L^{max}-L_k \end{array} \right) \quad (7)$$

ここに、 F : 目的関数増加の下限値。 y_{kl} : 0-1変数で、リンク k をリンクレベル $(L^{max}-l+1)$ から $(L^{max}-l)$ に下げるとき $y_{kl} = 1$ 、下げないとき $y_{kl} = 0$ 。 f_{kl} : リンク k をリンクレベル $(L^{max}-l+1)$ から $(L^{max}-l)$ に下げるときの目的関数増加の下限値。 C_{kl} : リンク k をリンクレベル $(L^{max}-l)$ から $(L^{max}-l+1)$ に改善するのに必要な費用。 C^{max} : すべてのリンクを最高のリンクレベル L^{max} に改善するのに必要な費用とする。

目的関数 F はリンクレベルを下げることによっておこる目的関数 F の増加の下限値の最小化を考えている。式(5)は式(2)と同値であり、右辺の $C^{max} - C^u$ はすべてのリンクのリンクレベルを最高にした状態から、これだけ以上は建設費を節約しないと式(2)を満たせないという値である。よって式(5)を満たす解は式(2)も満たすことになる。

式(6)はリンクレベルを下げる順序に関する制約である。

表-1 リンクレベル低下による目的関数増加の下限値

すなわちリンク l をリンクレベル $(L^{max}-l+1)$ から $(L^{max}-l)$ に下げない間は、リンクレベルを $(L^{max}-l)$ から $(L^{max}-l-1)$ には下げられないことを表す。式(7)は y_{le} が0-1変数であることを示す。

表-1はあるリンクのリンクレベルを1下げたときの目的関数増加の下限値 f_{le} に関する表である。リンク l については、現在のリンクレベル $L_l=0$ の場合を例にあげている。

リンクレベル	1	...	l	...	$L^{max}-L_l$	...	L^{max}
リンクレベル	$L \rightarrow (L-1)$	...	$(L-l+1) \rightarrow (L-l)$	...	$(L-l+1) \rightarrow (L-l)$	...	$1 \rightarrow 0$
1	f_{11}	...	f_{1l}	...	$f_{1, L^{max}-L_l}$	...	$f_{1, L^{max}}$
...	...	...	...	...	...	...	...
l	f_{l1}	...	f_{ll}	...	$f_{l, L^{max}-L_l}$	...	...
...	...	...	...	...	...	...	...
m	f_{m1}	...	f_{ml}	...	$f_{m, L^{max}-L_l}$	...	...

目的関数式(4)の f_{le} で表わされるリンク l の目的関数増加の下限値とは、リンク l をリンクレベル $(L^{max}-l+1)$ から $(L^{max}-l)$ に下げるときに、式(1)の目的関数 z が少なくともこれだけは増加するという値である。本研究では f_{le} を次のようにして求める。

いま、リンク l はノード j 、 j' を結んでいるとする。式(8)により、リンク l のリンクレベルが $(L^{max}-l+1)$ 、その他のリンクレベルがすべて L^{max} のときのノード j から j' への最短所要時間 $t_{jj'}^L$ を求める。ただし、この最短所要時間とは災害パターンごとに重みづけした所要時間の和である。

$$t_{jj'}^L = \sum_{s=1}^S w_s \cdot t_{jj'}^s(x^L, \phi_s) \quad (8)$$

同様にして式(9)により、リンク l のリンクレベルが $(L^{max}-l)$ 、その他のリンクレベルがすべて L^{max} のときの最短所要時間 $t_{jj'}^{L-1}$ を求める。

$$t_{jj'}^{L-1} = \sum_{s=1}^S w_s \cdot t_{jj'}^s(x^{L-1}, \phi_s) \quad (9)$$

$$\text{ただし、} x^L: \begin{cases} x_k = L^{max} & (k=1, 2, \dots, l-1, l+1, \dots, m) \\ x_k = L^{max}-l+1 & (k=l) \end{cases}$$

$$x^{L-1}: \begin{cases} x_k = L^{max} & (k=1, 2, \dots, l-1, l+1, \dots, m) \\ x_k = L^{max}-l & (k=l) \end{cases}$$

そして式(10)により、リンク l のリンクレベルを1下げたときのノード j から j' に到る所要時間の増分 $(t_{jj'}^{L-1} - t_{jj'}^L)$ に、ノード j から j' へのトリップ数 $g_{jj'}$ をかけた値として f_{le} を求める。

$$f_{le} = g_{jj'} \cdot (t_{jj'}^{L-1} - t_{jj'}^L) \quad (10)$$

式(4)~(7)の補助問題1は、複数の制約条件をもつ0-1整数線形計画問題であり、一般にその最適解を求めるには多くの計算が必要となる。

リンクを採用するかしないかのみを考える最適ネットワーク問題の場合は、式(6)のリンクレベルの低下順序に関する制約式に対応するものはない。このとき、Hoangは0-1整数線形計画問題から整数制約をはずしLP問題として、簡単に下限値を得る手法を提案している。Dantzig³⁾によりこのLP問題の最適解は、目的関数の係数を予算制約の係数で割り、その値の小さいものから予算制約を満足するまで順に1とおいいていき、最後の部分だけちょうど予算に一致する小数値とし、他の変数はすべて0として得られることが示されている。すなわち実際にシンプレックス法を使用することなく簡単に下限値を得ることができるのである。そこで、ここでもまず式(11)の整数制約を連続変数制約式(11)に置きかえ、LP問題とする。

$$0 \leq y_{le} \leq 1 \quad \begin{cases} l=1, 2, \dots, m \\ l=1, 2, \dots, L^{max}-L_l \end{cases} \quad (11)$$

これを補助問題2とし、目的関数を F とすれば、制約を緩和しているのだから、式(12)の関係が成立するのは明らかである。

$$F \geq F' \quad (12)$$

補助問題2にDantzigの解法を適用すると、制約式(6)を満たさない解が得られる場合が考えられる。そこで式

(6)を満たさなくなる変数を1つにましめることを考える。これが変換操作アルゴリズムである。

①すべての l について、式(13)を満たす l の集合 S_l を求める。

$$f_{l,l}/C_{l,l} \geq f_{l,l+1}/C_{l,l+1} \quad (13)$$

もしすべての l について、 $S_l = \emptyset$ なら変換終了。そうでなければ②へ。

②一度変換の終了したリンク番号の集合 $\bar{S}$ を空集合 $\emptyset$ とする。

③へ。

③式(13)を満たし、まだ変換を行っていない最小のリンク $\bar{l}$ を式(14)で求める。

$$\bar{l} = \min \{ l \mid S_l \neq \emptyset, l \notin \bar{S} \} \quad (14)$$

もし $\bar{l}$ が存在すれば④へ。しなければ①へ。

④式(15)により $\bar{l}$ を求め、この $\bar{l}$ 、 $\bar{l}$ について式(16)、(17)の変換を行い⑤へ。

$$\bar{l} = \min \{ l \mid l \in S_{\bar{l}} \} \quad (15)$$

$$f_{\bar{l}} \leftarrow f_{\bar{l},\bar{l}} + f_{\bar{l},\bar{l}+1} \quad (16)$$

$$C_{\bar{l}} \leftarrow C_{\bar{l},\bar{l}} + C_{\bar{l},\bar{l}+1} \quad (17)$$

⑤置き換えを行ったリンク $\bar{l}$ について、式(18)~(21)により添字 l の順序を整理し、⑥へ。

$$f_{\bar{l},\bar{l}} \leftarrow f_{\bar{l}} \quad (18)$$

$$C_{\bar{l},\bar{l}} \leftarrow C_{\bar{l}} \quad (19)$$

$$f_{\bar{l},l} \leftarrow f_{\bar{l},l+1} \quad (l = \bar{l}+1, \bar{l}+2, \dots, L - L_{\bar{l}}) \quad (20)$$

$$C_{\bar{l},l} \leftarrow C_{\bar{l},l+1} \quad (l = \bar{l}+1, \bar{l}+2, \dots, L - L_{\bar{l}}) \quad (21)$$

⑥ $\bar{S} \leftarrow \bar{l}$ として③へ

このアルゴリズムのフローチャートを図-3に示す。変換操作の例を図下表すと、図-4~図-6のようになる。ただし、これらの図はある1つの l についてのみ考えている。また変換を行うとき、添字 l が変わるので、区別するため l' で表すものとする。図-4の変換操作例1は、 $S_l = \emptyset$ の場合である。よって変換操作はしない。図-5の変換操作例2は、 $\bar{l} = 2$ の場合である。よって、式(16)~(19)の変換操作を行う。すなわち、 $f_{\bar{l}} \leftarrow f_{\bar{l},2} + f_{\bar{l},3}$ 、 $C_{\bar{l}} \leftarrow C_{\bar{l},2} + C_{\bar{l},3}$ として新しく、 $f_{\bar{l},2} \leftarrow f_{\bar{l}}$ 、 $C_{\bar{l},2} \leftarrow C_{\bar{l}}$ とする。

この状態で、 $S_{\bar{l}} = \emptyset$ となったのでこの l について変換終了。図-6の変換操作例3は操作を2回行う場合である。 $\bar{l} = 2$ なので、式(16)~(19)より、 $f_{\bar{l}} \leftarrow f_{\bar{l},2} + f_{\bar{l},3}$ 、 $C_{\bar{l}} \leftarrow C_{\bar{l},2} + C_{\bar{l},3}$ として、新しく $f_{\bar{l},2} \leftarrow f_{\bar{l}}$ 、 $C_{\bar{l},2} \leftarrow C_{\bar{l}}$ とする。この状態で再び $\bar{l} = 1$ となったので式(16)~(19)より、 $f_{\bar{l}} \leftarrow f_{\bar{l},1} + f_{\bar{l},2}$ 、 $C_{\bar{l}} \leftarrow C_{\bar{l},1} + C_{\bar{l},2}$ として新しく $f_{\bar{l},1} \leftarrow f_{\bar{l}}$ 、 $C_{\bar{l},1} \leftarrow C_{\bar{l}}$ とする。 $S_{\bar{l}} = \emptyset$ となったので、この l について変換終了。

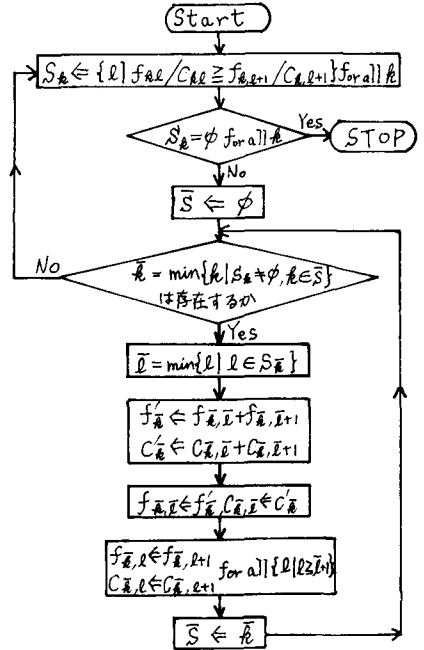


図-3 変換操作アルゴリズムのフローチャート

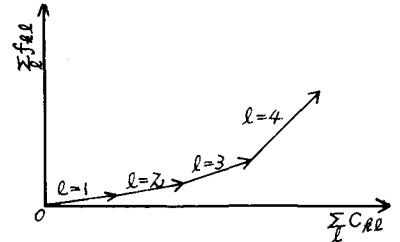


図-4 変換操作例 1

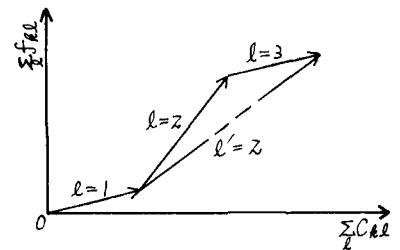


図-5 変換操作例 2

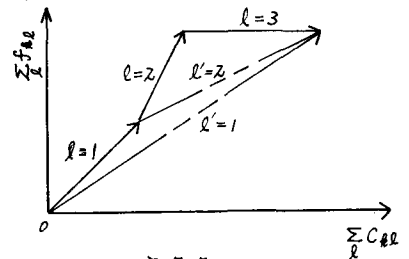


図-6 変換操作例 3

補助問題2に変換操作アルゴリズムを適用して得られる問題を補助問題3としてその定式化を次に示す。

補助問題3

$$\min F'' = \sum_{k=1}^m \sum_{l=1}^{\alpha_k} f_{kl} \cdot y_{kl} \quad (22)$$

$$\text{s.t.} \quad \sum_{k=1}^m \sum_{l=1}^{\alpha_k} C_{kl} \cdot y_{kl} \geq C^{\max} - C^u \quad (23)$$

$$0 \leq y_{kl} \leq 1 \quad \left(\begin{array}{l} k=1, 2, \dots, m \\ l=1, 2, \dots, \alpha_k \end{array} \right) \quad (24)$$

ここに、 α_k : 変換操作を行った後のリンク k についての l の上限値。 l : 変換操作を行った後の l 。補助問題1・2の l と区別するため l' とする。式(22)~(24)で表わされる補助問題3はナップザック問題の整数制約をはずした線形計画問題であり、Dantzigの簡単な解法を適用できる。次にその解法を示す。

$$f_{kl} / C_{kl} \quad \left(\begin{array}{l} k=1, 2, \dots, m \\ l'=1, 2, \dots, \alpha_k \end{array} \right)$$

を求め、その値の小さいものから順に、添字を1, 2, 3, ... と置きなおす。すると、補助問題3は式(25)~(27)の形で表わされる。

$$\min F''' = \sum_{k=1}^K f_{kl} \cdot y_{kl} \quad (25)$$

$$\text{s.t.} \quad \sum_{k=1}^K C_{kl} \cdot y_{kl} \geq C^{\max} - C^u \quad (26)$$

$$0 \leq y_{kl} \leq 1 \quad (k=1, 2, \dots, K) \quad (27)$$

ここに、 $K = \sum_{k=1}^m \alpha_k$ とする。

添字の決定方法により、式(28)が明らかに成立する。

$$f_1 / C_1 \leq f_2 / C_2 \leq \dots \leq f_K / C_K \leq \dots \leq f_K / C_K \quad (28)$$

このとき最適解は、式(29)~(31)となる。

$$\left\{ \begin{array}{ll} y_{kl} = 1 & \text{if } k < \nu \\ y_{kl} = 0 & \text{if } k > \nu \end{array} \right. \quad (29)$$

$$\left\{ \begin{array}{ll} y_{kl} = 0 & \text{if } k > \nu \\ y_{kl} = (C^{\max} - C^u - \sum_{k=1}^{\nu-1} C_{kl}) / C_{\nu} & \text{if } k = \nu \end{array} \right. \quad (30)$$

$$y_{kl} = (C^{\max} - C^u - \sum_{k=1}^{\nu-1} C_{kl}) / C_{\nu} \quad (31)$$

ただし、 ν は式(32)を満たす最小の整数である。

$$\sum_{k=1}^{\nu} C_{kl} \geq C^{\max} - C^u \quad (32)$$

図-5, 6を見てわかるように、2点鎖線のベクトルで表わされている変換操作を行った部分は、実線のベクトルで表わされている部分よりも必ず下方に位置している。このことは式(33)が成立することを意味する。

$$F' \geq F'' \quad (33)$$

またDantzigの解法は補助問題3の厳密解を与えるのだから式(34)が成立する。

$$F'' = F''' \quad (34)$$

すなわち式(12), (33), (34)より式(35)が成立する。

$$F \geq F'' = F''' \quad (35)$$

よって式(36)で表わされる Z^L は、目的関数の下限値となる。

$$Z^L = Z^0 + F'' \quad (36)$$

ここに、 Z^0 : すべてのリンクレベルを $L^{\max}$ としたときの目的関数 Z の値とする。

以上述べてきたことをまとめて、次のアルゴリズム1を作成する。

(3) アルゴリズム1とその組合せトリ－

最初すべてのリンクを最高のリンクレベルにしておき、順番にリンクレベルを下げるにより実行可能解を探索する。この探索には組合せトリ－を使用するが、分枝変数は、補助問題1で導入した y_{kl} を使う。

$y_{kl}=1$ はリンク k のリンクレベルを $(L^{\max} - l + 1)$ から $(L^{\max} - l)$ に下げること意味する。 $y_{kl}=0$ はリンク

れのリソクレベルを $(L^{\max}-l+1)$ に固定し、それ以上は下げないことを意味する。 $y_{kl}=-1$ は y_{kl} が自由変数であることを表す。組合せトリーの一部を図-7に示す。節点を①, ..., (I+2), ⑦で表す。節点Iから分枝変数 y_{kl} を1とにおいて分枝し、節点(I+1)に進み、さらに y_{kl} を1とにおいて分枝し、節点(I+2)に進んでいる。このことは、まずリンク l_1 のリソクレベルを $(L^{\max}-l_1+1)$ から1レベル下げ、さらにリンク l_2 のリソクレベルを $(L^{\max}-l_2+1)$ から1レベル下げることを意味する。節点(I+2)以下の節点をすべて探索した後、再び節点(I+2)に戻り、次に y_{kl} を0とにおいて節点⑦に進む。

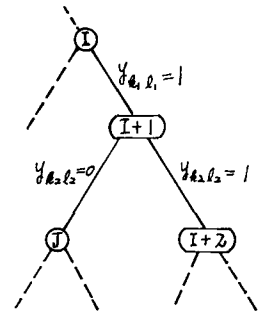


図-7 アルゴリズム1の組合せトリー

すなわち backtracking を行う。このことは、リンク l_1 のリソクレベルを $(L^{\max}-l_1+1)$ に固定することを意味し、以下の節点ではリンク l_2 のリソクレベルが $(L^{\max}-l_2+1)$ であるとして、他の自由変数をもつリンクのリソクレベルの組合せを探索することになる。

組合せトリー上のある節点の解 X は式(37)により、分枝変数 $y_{kl}=1$ である部分だけを最高のリソクレベル $L^{\max}$ から下げて得られる。

$$z_k = L^{\max} - \sum_{l=1}^{L^{\max}} \varphi(y_{kl}) \quad (k=1, 2, \dots, m) \quad (37)$$

$$\text{ここに、} \begin{cases} \varphi(y_{kl}) = 1 & \text{if } y_{kl} = 1 \\ \varphi(y_{kl}) = 0 & \text{if } y_{kl} \neq 1 \end{cases} \quad \text{とする。}$$

アルゴリズムの中で、 z^{l_1}, z^{l_2} の2つの下限値を使用する。 z^{l_1} は補助問題3のLP解から式(36)で求まる値である。このLP解は変換操作により合成された部分それぞれ1つの変数に置きかえたときの解であるから、分枝する際には補助問題1を使用しているもの y_{kl} になおして、同一リンクに対しては順番にリソクレベルが下がるようにする。そしてLP解の小数部分については、予算制約を満たした時点で、分枝を停止する。そしてその節点に対応する実行可能解の目的関数の下限値を z^{l_2} とする。

- ①初期値として、節点①で最適解の目的関数 z の値を $z_{\text{opt}} = \infty$ 、コストを $C_{\text{opt}} = \infty$ とする。すべての l について、分枝変数は $y_{kl} = -1$ とにおいて自由変数とする。②へ。
- ②すべてのリンクについて、リソクレベルを $L^{\max}$ としたときの目的関数 z の値を求め、 z とする。式(10)を使いすべての l について f_{kl} を求め補助問題2を作成する。③へ。
- ③補助問題2に変換操作アルゴリズムを適用し、補助問題3を作成する。④へ。
- ④その節点で固定変数を制約に加え、補助問題3をDantzigの解法を使い解く。実行不能なら⑧へ。実行可能なら式(36)により目的関数の下限値 z^{l_1} を求める。もし $z^{l_1} > z_{\text{opt}}$ なら⑧へ。他は⑤へ。
- ⑤補助問題3の解を使い、補助問題1の実行可能解が得られるまで $y_{kl} = 1$ とにおいて順番に分枝する。このときの下限値 z^{l_2} を求める。もし $z^{l_2} > z_{\text{opt}}$ なら⑦へ。他は⑥へ。
- ⑥その節点の解 X と目的関数 z の値を求め $z \leftarrow z$ とする。またそのときのコストを C とする。そして (z, C) が $(z_{\text{opt}}, C_{\text{opt}})$ より辞書編集的に小さければ $(z_{\text{opt}}, C_{\text{opt}}) \leftarrow (z, C)$ でおきかえ、最適解を $X_{\text{opt}} = X$ とする。⑦へ。
- ⑦その節点に来るとき分枝変数 y_{kl} の値を0に固定した節点に進む。④へ。
- ⑧その節点から分枝変数 y_{kl} が1である節点まで分枝変数 y_{kl} を-1、すなわち自由変数におきなおしながら、backtrackする。①へ。もどる節点がない場合は計算終了で、そのときの X_{opt} が最適解である。

(4) 簡単な計算例を使用したアルゴリズムの説明

簡単な例題として図-8のネットワークを考える。図中リンクの上にある数字はリンク番号を表す。表-2はOD表である。表-3にリンク長、改良前のリソクレベル、改良に要する費用、災害パターンとその重みを示

す。災害パターンは3つ考え、それぞれの重みは $w_1=0.7$, $w_2=0.2$, $w_3=0.1$ としている。パスが存在しないノード間については、所要時間のかわりにペナルティとして15を与えることにする。ここでは考える最高のリンクレベルを3とする。

すべてのリンクレベルを3にするために必要なコストは $C^{\max}=86$ である。このとき目的関数は $Z^0=145$ となる。

目的関数増加の下限値 f_{rel} を求めると表-4のようになる。現在のリンクレベルより下がることはないの、その部分には値を入れていない。表-4の f_{rel} を表-3の C_{rel} で割り、表-5を得る。さらに変換操作を行い、表-6の $f_{\text{rel}}/C_{\text{rel}}$ を得る。表-6で-1とあるのはその部分を変換していることを示す。

予算制約を $C^u=56$ とすると、図-10の組み合わせトリ-は次のようにして作成される。

① 節点0で、 $y_{21}=y_{31}=y_{41}=1$ と固定しても目的関数 Z の値は Z^0 より大きくならない。よってこの状態から計算を開始する。補助問題3を解いて $y_{32}=y_{41}=y_{32}=1$, $F''=0.5 \times 1 + 1.5 \times 1 + 3 \times 1 = 5$, $Z^1=145 + 5 = 150 < Z_{\text{opt}} = \infty$ 。よって分枝する。

② 節点3で、 $y_{32}=y_{41}=y_{32}=1$ で補助問題1が実行可能。 $Z^2=145 + 0.5 \times 1 + 1.5 \times 1 + 3 \times 1 = 150 < Z_{\text{opt}} = \infty$ 。よって目的関数を計算する。このときの解は、 $X=(2, 2, 1, 3, 1)$ である。図-9 (a)~(c)は解 X に災害パターン1, 2, 3を与えたときのネットワークの状態を示す。それぞれの総所要時間は145, 160, 170であるので、さらに重みづけして $Z=0.7 \times 145 + 0.2 \times 160 + 0.1 \times 170 = 150.5$ を得る。 $Z < Z_{\text{opt}} = \infty$ なので、この解を暫定解とし、 $X_{\text{opt}}=(2, 2, 1, 3, 1)$, $Z_{\text{opt}}=150.5$ とする。backtrackし、 $y_{32}=0$ と固定する。

③ 節点4で、固定変数 $y_{21}=y_{31}=y_{41}=y_{32}=y_{41}=1$, $y_{32}=0$ の下で補助問題3を解いて $y_{32}=1$, $y_{41}=2/20$, $F''=0.5 \times 1 + 1.5 \times 1 + 3.0 \times 1 + 18.0 \times 2/20 = 6.8$, $Z^1=145 + 6.8 = 151.8 > Z_{\text{opt}}$ 。よって節点5にbacktrackする。このとき $y_{32}=-1$, $y_{41}=0$ とする。同様の計算を繰り返し、節点12からbacktrackした際に、節点0にもどり、それ以上backtrackできなくなる。よってそこで計算を終了し、最適解 $X_{\text{opt}}=(2, 2, 1, 3, 1)$

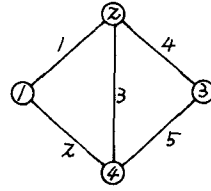


図-8 例題1のネットワーク

表-2 OD表

OD	1	2	3	4
0	1	2	3	4
1		15	5	10
2			20	15
3				5
4				

表-3 計算例の諸データ

リンク 番号	リンク 長	リンク レベル	改良費用			災害パターン		
			0→1	1→2	2→3	$w_1=0.7$	$w_2=0.2$	$w_3=0.1$
1	3	2	—	—	5	0	1	2
2	2	1	—	6	5	0	0	1
3	2	0	10	8	5	0	1	0
4	1	0	20	10	10	0	2	1
5	2	1	—	2	5	0	0	1

表-4 目的関数増加の下限値 f_{rel} 表-5 $f_{\text{rel}}/C_{\text{rel}}$

リンク番号	リンクレベル	3→2	2→1	1→0
1	1.5	—	—	—
2	0.0	3.0	—	—
3	0.0	3.0	12.0	—
4	12.0	6.0	42.0	—
5	0.0	0.5	—	—

リンク番号	リンクレベル	3→2	2→1	1→0
1	0.3	—	—	—
2	0.0	0.5	—	—
3	0.0	0.375	1.2	—
4	1.2	0.6	2.1	—
5	0.0	0.25	—	—

表-6 $f_{\text{rel}}/C_{\text{rel}}$

リンク番号	リンクレベル	3→2	2→1	1→0
1	0.3	—	—	—
2	0.0	0.5	—	—
3	0.0	0.375	1.2	—
4	0.9	-1	2.1	—
5	0.0	0.25	—	—

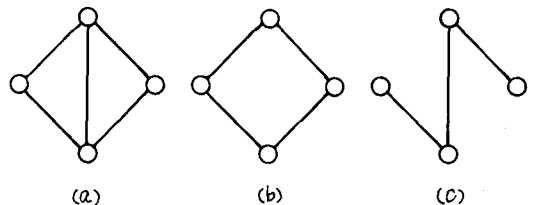


図-9 各災害パターンに対するネットワークの状態

を得る。ただし図-10で◎は目的関数を実際に計算した節点、○は $Z^L > Z_{opt}$ となった節点、○は $Z^L > Z_{opt}$ となった節点を示す。この例題においては、最初の実行可能解が最適解であり、目的関数の計算は1回だけでよかった。

(5) 電子計算機を使用した計算例

上記のアルゴリズム1による電子計算機のためのプログラムを作成した。そして図-11に示す、ノード数8、リンク数20のネットワークについて実際に計算を行った。図中、改良前のリンクレベルはすべて0として破線で表わしている。表-7はOD表で、乱数を使用して決めたものである。この例題では、リンクに向きはないものとしているので、三角表となっている。また災害パターンは図-12(a)~(d)に示す4種類を考えている。災害強度レベルが高いほどリンクを太くした。破線は災害強度レベルが0であることを表わしている。

最高のリンクレベルを5としたときの最適解を図-13(a)~(c)に示す。(a)~(c)はそれぞれ予算の上限値を1900, 1500, 1100, と小さくしていったときの状態を表わす。リンクの太さはリンクレベルに対応している。この図から重要なリンクで、災害強度レベルの強いものは、よりリンクレベルの高いものに改善されていることがわかる。

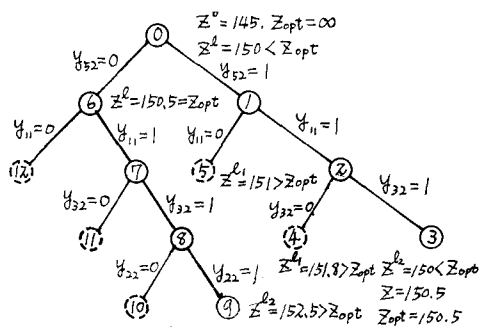


図-10 計算例の組み合せトリー

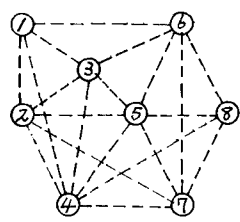


図-11 改良前のネットワーク

表-7 OD表

D \ O	1	2	3	4	5	6	7	8
0								
1		70	97	31	30	98	0	74
2			63	48	20	13	12	57
3				25	10	78	99	38
4					45	75	44	37
5						59	94	63
6							26	9
7								28
8								

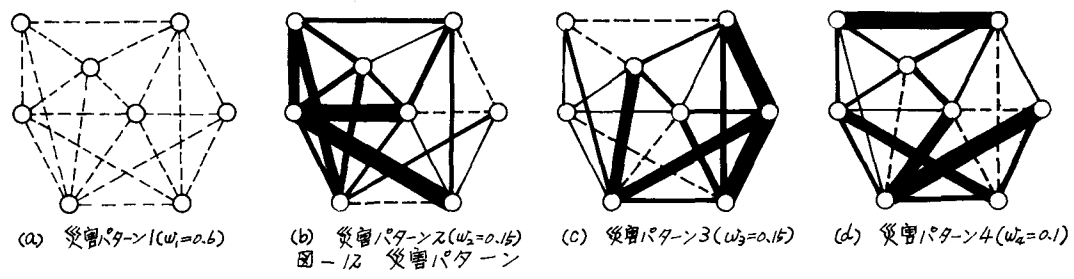


図-12 災害パターン

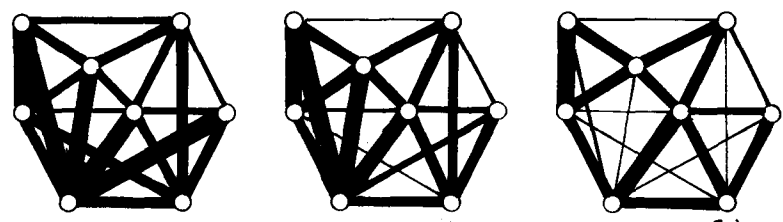


図-13 いくつかの予算制約に対する最適解

4 到達トリップ数最大化問題

(1) 問題の定式化

いくつかの災害パターンのもとでの目的地に到着可能なトリップ数の期待値を評価関数とする。制約条件は、の場合と同様に建設費とすると、この問題は次のように定式化される。

$$\max Z_2 = \sum_{i,j} W_{ij} \left(\sum_{\omega} \sum_{\phi} g_{ij} \cdot r_{ij}(\omega, \phi) \right) \quad (38)$$

$$\text{s.t.} \quad \sum_{k=1}^m C_k(x_k) \leq C^u \quad (39)$$

$$L_k \leq x_k \leq L_k^{\max} \quad (k=1, 2, \dots, m) \quad (40)$$

ここに、 Z_2 : 目的関数で、目的地に到達可能なトリップ数の期待値。 r_{ij} : ノード i からノード j に到達可能な確率。その他の変数については、3. の定義と同様とする。

式(38)の $g_{ij} \cdot r_{ij}(\omega, \phi)$ はリンクレベルが ω 、災害パターンが ϕ のとき、ノード i からノード j に到達可能なトリップ数の期待値である。それをすべてのノード対について合計し、さらに災害パターンごとに重みづけしたものが Z_2 である。すなわち、目的関数 Z_2 ではトリップの所要時間は評価に考慮されていないことになる。制約式(39)、(40)は式(2)、(3)と同様である。

(2) 信頼度の下限値の導入

ノード i, j 間の厳密な信頼度 r_{ij} を得ることは困難であるので、ここではカットセットにより信頼度の下限値を求め、これを式(38)の r_{ij} のかわりに使用する。

ここにいうカットセットとは、あるノード対に対して存在するリンクの集合で、このリンクがすべて破壊すれば他のリンクの状態に関係なく、考えているノード間にパスがなくなるものである。たとえば、図-14のネットワークにおいてノード1とノード4の間に4つのカットセットが存在する。それらを表-8に示す。

信頼度の下限値 r_{ij} は式(41)により求められる。⁴⁾

$$r_{ij} = \prod_{v \in V_{ij}} [1 - \prod_{k \in P_v} P_k] \quad (41)$$

ここに、 r_{ij} : 信頼度の下限値。 V_{ij} : ノード i, j 間のカットセットの番号の集合。 v : カットセットの番号。 P_v : カットセット v に含まれるリンク番号の集合。

k : リンク番号。 P_k : リンク k の破壊確率とする。

よって目的関数式(38)は、式(42)で表わされる。

$$\max Z_3 = \sum_{i,j} W_{ij} \left(\sum_{\omega} \sum_{\phi} g_{ij} \cdot r_{ij}(\omega, \phi) \right) \quad (42)$$

(3) アルゴリズムとその組合せトリー

カットセットの集合 V_{ij} を求めさえすれば、ノード対 i, j 間のパスの存在する確率の下限値 r_{ij} を式(41)により簡単に求めることができる。したがって式(42)の目的関数 Z_3 の値はすべてのノード対についてトリップ数 g_{ij} と r_{ij} の積の和をとることにより容易に求めるようになった。よって最適解は、予算制約を満たすすべての実行可能解を、組合せトリーを使って列挙し、暫定解を更新してゆけば求めることができる。

分枝変数としてはリンク k のリンクレベルを表わす x_k を使用する。 $x_k = -1$ で表わされる x_k は自由変数で、まだリンク k のリンクレベルが決っていないことを意味する。

組合せトリーの一部を図-15に示す。節点①、---

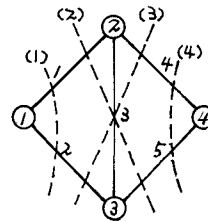


図-14 カットセットの例

表-8 図-12のカットセット

カットセット	リンク
(1)	1, 2
(2)	1, 3, 5
(3)	2, 3, 4
(4)	4, 5

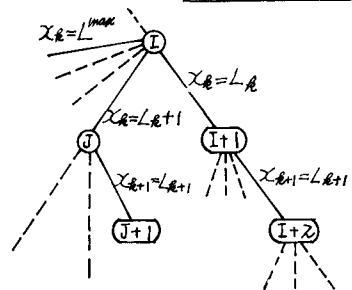


図-15 アルゴリズムZの組合せトリー

(I+2), ①, (J+1)で表れる。節点Iから分枝変数 x_k を現在のリンクレベル L_k と分枝し、節点(I+1)に進む。さらに x_k を現在のリンクレベル L_{k+1} と分枝し、節点(I+2)に進んでいる。分枝変数を選ぶ順序は任意であるので、ここではリンク番号の順に従うとしている。節点(I+1)以下の節点をすべて探索した後再び節点(I+1)にもどってくる。そして次に x_k をもし可能なら L_{k+1} と分枝した節点Jに進み、以下他の自由変数をもつリンクのリンクレベルの組合せを探索する。Jでの組合せトリーでは分枝変数が一部自由変数のままの状態であっても予算制約を満たしたときの節点で、式(37)より実行可能解が得られた。しかしここではすべての分枝変数が固定されたときの節点が1つの実行可能解を表わすことになる。以下にそのアルゴリズムを提案する。

まず、すべてのノード対 ij についてカットセットの集合 V_{ij} を求めておく。目的関数の最大化問題であるから、初期値として暫定解の目的関数値を $-\infty$ 、コストを ∞ とおき、すべての分枝変数を自由変数として計算を開始する。最初、すべてのリンクのリンクレベルが現在のリンクレベルである節点に分枝して進む。このとき当然コストは0となり1つの実行可能解が求まることになる。これを暫定解とする。そして最後に分枝したリンクについて、順番にリンクレベルを上げていきつつ目的関数値、コストを比較して暫定解を更新していく。コストが制約を上まわるか、最高のリンクレベルを越えれば、そのリンクのリンクレベルを自由変数におきなおして、さらに1つ上の節点にbacktrackし、そのリンクについて同様の操作を繰り返しながらすべての解の組合せを探索する。

アルゴリズムを整理すると次のようになる。

①カットセットの集合 V_{ij} を求める。②へ。

②初期値 $Z_{opt} = -\infty$, $C_{opt} = \infty$, X_{opt} を与える。 $x_k = -1$ ($k=1, 2, \dots, m$)とする。③へ。

③ $x_k = -1$ であるリンクが残っていれば④へ。なければ⑤へ。

④リンク k のリンクレベル x_k を $x_k = L_k$ として分枝する。③へ。

⑤その節点での実行可能解 X に対する目的関数とコストを求め Z , C とする。 $Z > Z_{opt}$ のとき、または $Z = Z_{opt}$ でかつ $C < C_{opt}$ のとき Z_{opt} , C_{opt} , X_{opt} を Z , C , X で置き換える。⑥へ。

⑥最後に分枝した変数に対応するリンクのリンクレベルを1上げた節点へ進む。⑦へ。

⑦リンクレベルを上げたときに、最高のリンクレベルを越えれば④へ。最高のリンクレベル以下であれば⑧へ。

⑧コストが制約以内であれば③へ、制約を越えれば④へ。

⑨分枝変数に対応するリンク k のリンクレベル x_k を-1としてさらに手前の節点にもどり、その節点へのリンクレベルを1上げた節点へ進む⑦へ。もどる節点がない場合は計算終了。そのときの X_{opt} が最適解である。

ここに、 X :目的関数を計算したときの各リンクのリンクレベル。 L_k :現在のリンクレベルとする。

アルゴリズムのための電子計算機プログラムも作成している。又、3の計算例を解き、アルゴリズムの実証を行った。ただし、この方法ではすべての実行可能解を列挙するため、あまり大規模なネットワークには適用できない。

(4) 対数を使用した評価関数の導入

目的関数式(42)は、各ノード対間にパスの存在する確率の下限値をトリップ数で重みづけした形になっているため、トリップ数の大きいノード対間の信頼性改善を重視していることになる。そのため、もしノード対 ij 間のトリップ数が小さければ、 ij 間の信頼度は改善される機会が少なくなる。

そこで、たとえ ij 間のトリップ数が小さくても、 ij 間の信頼度が極端に低い場合にその信頼度を改善するような、すなわちすべてのノード対間にある程度の信頼度を確保しつつ、ネットワーク全体の最適化をはかるような評価関数として、対数を使用した式(43)を導入する。

$$\max Z_4 = \sum_{i=1}^n \sum_{j=1}^n w_{ij} \left(\sum_{i=1}^n \sum_{j=1}^n g_{ij} \cdot \log r_{ij}(x, \phi_s) \right) \quad (43)$$

(5) アルゴリズム3の提案

式(39),(40)の制約下で、式(43)の z_4 を最大にする問題の解法について述べる。3での総所要時間を評価関数としたときの最適化手法と同様にして、計算の効率化をはかる。この問題は目的関数の最大化問題なので、目的関数の上限値を設定する。このため次の問題を導入する。

$$\min G = \sum_{k=1}^m \sum_{l=1}^{L_k^{\max}} g_{kl} \cdot y_{kl} \quad (44)$$

$$\text{s.t.} \quad \sum_{k=1}^m \sum_{l=1}^{L_k^{\max}} C_{kl} \cdot y_{kl} \geq C^{\max} - C^u \quad (45)$$

$$y_{kl} \geq y_{k,l+1} \quad \left(\begin{array}{l} k=1, 2, \dots, m \\ l=1, 2, \dots, L_k^{\max} - L_k \end{array} \right) \quad (46)$$

$$y_{kl} = 0 \text{ or } 1 \quad \left(\begin{array}{l} k=1, 2, \dots, m \\ l=1, 2, \dots, L_k^{\max} - L_k \end{array} \right) \quad (47)$$

ここに、 G : 目的関数 z_4 の減少の下限値。 g_{kl} : リンク k をリンクレベル $(L_k^{\max} - l + 1)$ から $(L_k^{\max} - l)$ に下げたときの目的関数 z_4 の減少の下限値。他の変数については、3の補助問題1と同様である。

目的関数式(44)は目的関数減少の下限値の最小化を考えている。制約式(45)~(47)は式(5)~(7)と同じである。式(44)の g_{kl} の求め方については、次のようにする。

まず、リンク k のリンクレベルを $(L_k^{\max} - l + 1)$ にし、他のすべてのリンクレベルを $L_k^{\max}$ にしたときの目的関数 z_4 を求める。そしてリンク k のリンクレベルを $(L_k^{\max} - l)$ にし、他のすべてのリンクレベルを $L_k^{\max}$ にしたときの目的関数 z_4 の値を求め、さきに求めた値からこの値を差し引く。これが g_{kl} である。

式(44)~(47)で表わされる問題は、3での補助問題1と同じ形になっている。よってこのときと同じ手法で、整数制約をはずし、さらに変換操作アルゴリズムを適用したのち、Dantzigの解法を使用する。このときの値を F に対応させて G' とすると、目的関数 z_4 の上限値 z^u は、式(48)で求めることができる。

$$z^u = z^0 - G' \quad (48)$$

ここに、 z^0 : すべてのリンクを最高リンクレベルにしたときの目的関数値とする。

基本的な解法はアルゴリズム1と同じである。すなわち、最初すべてのリンクを最高のリンクレベルにしており、順番にリンクレベルを下げることにより実行可能解を探索する。組合せトリーを作るときの分枝変数は、 y_{kl} を使う。アルゴリズム1と異なるのは、この問題が目的関数の最大化問題であるため、目的関数の上限値が暫定解の目的関数値未満であれば目的関数の計算を省略できる点である。また実際に分枝した節点での目的関数 z_4 の上限値を z^u とする。アルゴリズム3を整理すると以下のようになる。

- ①初期値として、節点0で最適解の目的関数 z_4 の値を $z_{opt} = -\infty$ 、コストを $C_{opt} = \infty$ とする。すべての k, l について分枝変数は $y_{kl} = -1$ と置いて自由変数とする。②へ。
- ②すべてのリンクについてリンクレベルを $L_k^{\max}$ としたときの目的関数 z_4 の値を求め z^0 とする。すべての k, l について g_{kl} を求め、式(44)~(47)で表わされる問題を作成する。③へ。
- ③変換操作アルゴリズムを適用し、新たにLP問題を作成する。④へ。
- ④その節点で固定変数を制約に加え、LP問題を解く。実行不能なら⑧へ。実行可能なら、式(48)により目的関数の上限値 z^u を求める。もし $z^u < z_{opt}$ なら⑧へ。他は⑤へ。
- ⑤LP問題の解を使い、式(45)を満たす実行可能解が得られるまで $y_{kl} = 1$ と置いて順番に分枝する。このときの上限値 z^u を求める。もし $z^u < z_{opt}$ なら⑧へ。他は⑥へ。
- ⑥その節点の解 X と目的関数 z_4 の値を求め $z \leftarrow z_4$ とする。またそのときのコストを C とする。そして $z > z_{opt}$ あるいは、 $z = z_{opt}$ かつ $C < C_{opt}$ であれば、 $z_{opt}, C_{opt}, X_{opt}$ を z, C, X で置き換える。⑦へ。
- ⑦その節点にくるときの分枝変数 y_{kl} の値を0に固定した節点に進む。④へ。
- ⑧その節点から分枝変数 y_{kl} が1である節点まで、分枝変数 y_{kl} を-1とおきおしながら backtrack する。

⑦へ。もどる節点がない場合は計算終了下、そのときの X_{opt} が最適解である。

このアルゴリズム3のための電子計算機プログラムを作成した。アルゴリズム2に比べれば不要な計算が省略できる下、より効率的な計算が可能となった。

5 あとがき

本研究では災害に対して信頼性の高い交通ネットワークを合理的に構成するアルゴリズムを開発した。

まず、交通ネットワークの特性を検討した上下、この問題を2つの場合に分けて定式化した。ひとつは総所要時間最小化問題で、他のひとつは目的地に到着可能な期待トリップ数最大化問題である。

われわれは、定式化する際、いくつかの新しい概念を導入した。まず、リンクの規格のようにみなされ、対災害信頼性の水準を離散的に表わすものとしてリンクレベルを設定した。そして簡単化のために災害の強度を離散的に災害強度レベルで表わした。このことにより、本研究で取り扱う問題は一定の予算制約下で上述の評価関数を最小あるいは最大にするようなリンクレベルを探索する組合せ問題に帰着された。

総所要時間最小化問題については、目的関数の下限値を設定することにより、効率的に計算できるアルゴリズム1を作成した。

目的地に到着可能な期待トリップ数最大化問題については、カットセットの概念を導入することにより、期待トリップ数の下限値を容易に求めることのできるアルゴリズム2を作成した。さらに、ネットワークで部分的に極端に信頼度の低いノード対がなくなるよう、対数を使用した評価関数を新たに設定した。そして目的関数の上限値を求めることにより、効率的に計算できるアルゴリズム3を作成した。

それぞれのアルゴリズムのプログラムを作成し、実際にいくつかの例題を解いた結果、とくに目的関数の下限値あるいは上限値を使用したアルゴリズム1、3が有効であることがわかった。

ここで開発したアルゴリズムはすべて厳密解法であるが、実用性を重視した近似解法もこれをもとに考えられよう。従来の最適ネットワーク問題がそうであるように、実際の問題に適用するには近似解法でなければならない。そのような近似解法をつくり、実際の問題への適用を図ることが、今後の課題である。

参考文献

- 1) Hoang Hai Hoc : A computational approach to the selection of an optimal network. Management Science, Vol 19, No. 5, PP. 488~498, January, 1973.
- 2) 枝村 俊郎・森津秀夫 : 最適交通ネットワーク問題の厳密解法と近似解法. 土木学会論文報告集, 才262号, PP. 39~53, 1977年6月.
- 3) Dantzig, G. B. : Discrete-variable extremum problems. Operations Research, Vol. 5, PP. 266~277, 1957.
- 4) Jensen, P. A. and M. Bellmore : An algorithm to determine the reliability of a complex system. IEEE Transactions on Reliability Vol. R-18, PP. 169~174, November, 1969.
- 5) 枝村俊郎・森津秀夫・土井元治 : 対災害信頼性による交通網計画. 土木学会才34回年次学術講演会講演概要集, 才4部, PP. 53~54, 森津秀夫・土井元治・中川勝八郎 : 破壊確率を考慮した対災害信頼性の交通網への最適配分. 土木学会才34回年次学術講演会講演概要集, 才4部, PP. 55~56, 昭和54年10月.