1．科学技術計算におけるソフトウェアの生産性について

日本原子力研　浅井　清，富山峯秀
Kiyoshi Asai　Minehide Tomiyama
富士通(株)　田子精男，松浦俊彦
Yoshio Tago　Toshihiko Matsuura

はじめに

大きなソフトウェア開発プロジェクトをいくつか経験すればソフトウェアの生産性は見当がつくといわれているが科学技術計算の場合はどうであろうか．

筆者らの研究所でも技術開発に関連して計算機プログラムが多数使用されており，それらのかなりのものは外注形式で作成されている．プログラム開発についてのこの方式は今後ますます強まる傾向にあるので，ソフトウェアの生産性に関して一定の基準を持つことが必要となってきた．これがこの報告を作成した動機のひとつである．

1．使用したデータと評価結果

日本原子力研究所計算センタに富士通（株）より派遣された外来研究員約10名が昭和52年度，53年度に変換したプログラム38本，作成したプログラム20本を対象データとした．外来研究員の本来の職務は外国で開発された計算機プログラムの変換作業である．したがって作成作業よりは変換作業が多くなっている．

変換したプログラムの環境条件を数値情報に置き換える過程でいくつかのプログラムが単純な変換ではなく，改良と呼ぶべきものであることがわかった．

作成したプログラムの環境条件を数値情報に置き換える過程でいくつかのプログラムが定型的なプログラミングで作られたものではなく，問題分析作業の結果であることがわかった．

そして問題分析，プログラミング，単純変換をグラフ上でプロットしてみると，それぞれのグループがまとまっており，またグループ間に共通な部分がほとんど見いだせないことがわかった．

またプログラムの改良は上記三要素の組み合せである．

三要素について簡単に述べておこう．

(1)問題分析

これはむずかしい問題を計算機プログラム化しようとする場合で，研究開発機関に典型的な例である．その詳細については附録で述べる．

(2)定型的プログラミング

これは仕様書の指定通りにプログラムを作成するときの生産性を示す．採用された解法は，対象となるデータに対して正しい答を出すことがわかっている．

(3)単純変換

これはすでに完成しているプログラム，またはデータにわずかな変更をほどこすときの生産性を示す．

筆者らの研究所では外国から入ってきた原子力コード（プログラムとデータ）を研究所の計算機に適用しようとする変換が多い．

三要素の生産性（作成されたソース・カード枚数，月単位）はつぎの表-1のようになる．表-1に対応するグラフは図－1，2である．

表-1　三要素の生産性

問題分析	プログラミング	単純変換
$L=151M^{1.4}$	$L=759M^{0.64}$	$L=3311M^{0.55}$
$L=98e^{0.057x}$	$L=686e^{0.023x}$	$L=182be^{0.048x}$

注(1) Lは生産されるソース・カード枚数，Mは月数，xは生産性指標を示す．eは自然対数の底．
(2) 対象としたデータの性質から，問題分析の式ではMを3以上に外挿してはならず，他の場合は10を越えてMを外挿してはならない．

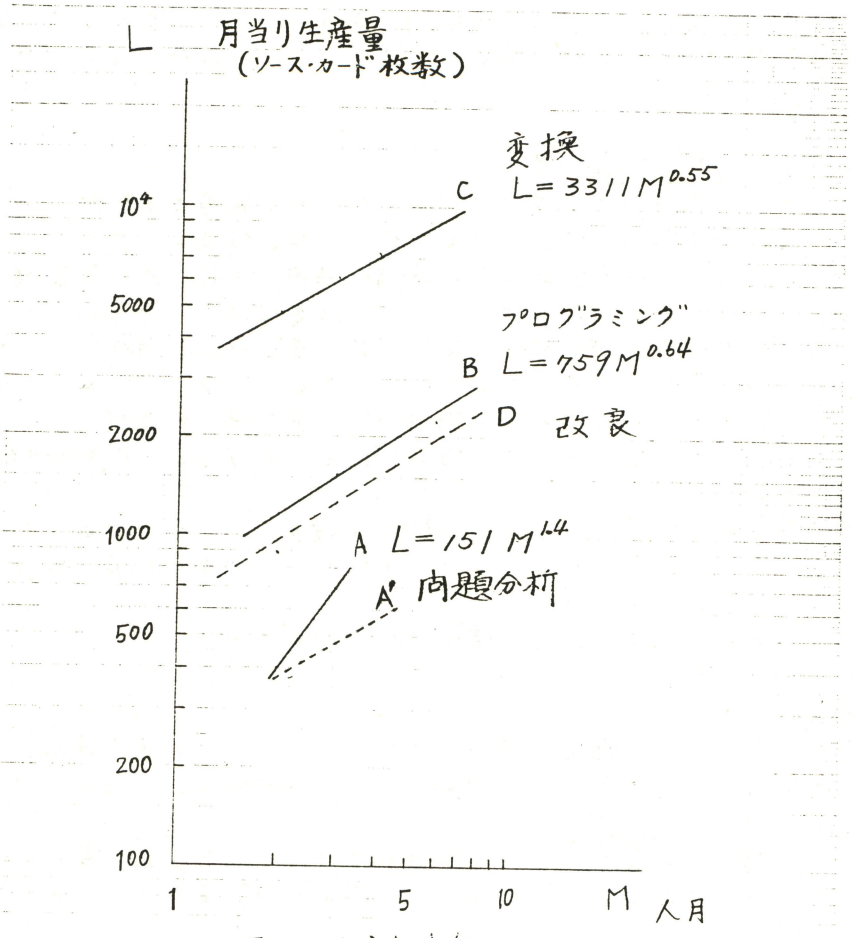

図-1　生産性直線

注(1) 直線Aは附録で述べているごとく点線A'になるはずである．

2. Walston and Felix の推定法

計算機プログラム開発費用の推定は，開発前の環境条件をみることによって予測する方法と，できあがった製品から推定する方法の2種がある．

Walston and Felix 法は前者に属し[1,2]，その特徴は，環境条件を数値情報に変換し，これらの数値の和として生化性指標なるものを定義することである．

Walston と Felix はプログラムの生産性に影響あると考えられる項目を29ケ列挙し，各項目を A；条件のよいとき，B；普通，C；条件の悪いときの3つの場合に分類した．そしてそれらの各項目がそれぞれ独立に生産性に影響をおよぼすと仮定し，その影響を約60のソフトウェア・プロジェクトについて調査した．項目 i について条件のよいときに生産されるプログラム・ステップ数を Ai，悪いときに生産されるプログラム・ステップ数を Ci とすると，生産性の変化 Si は $Si = Ai - Ci$ であらわされる．また生産性指標 I は

$$I = \sum_{i=1}^{N} Wi\, Xi$$

で計算される．

ここで I はひとつのプログラムの作成，または変換についての生産性指標，
Wi は $(1/2)\log_{10} Si$，
Xi は 調査表の回答が A なら +1，B なら 0，C なら −1 の値をとる，
N は 調査対象となった項目数である．

前節1．で述べた考え方にそって筆者らは問題分析，定型的プログラミング，単純変換の諸作業について別々の調査表を作成した．それらの項目は Walston and Felix のそれにならっているが，調査表の種類によって少しずつ異っている．ここでは定型的プログラミング作業に関する調査表の項目を表-2に例示する．丸印のついている項目は $Ai - Ci$ の値が有意（正の値）となり，生産性指標の計算に使用可能となったものである．

表-2　調査項目（定型的プログラミング）

複雑さの要素	○発注者，○要求定義時の利用者の参加， ○発注者の検証能力
発注者にかかわる要素	○当該応用分野での経験，計算機分野での経験， ○プログラマへの協力，○仕様書の完成度， ○コード設計書の完成度，○テスト・データ提供， 計算センタ運用の知識，当該計算機の知識， ○仕様の変更
プログラマにかかわる要素	○当該応用分野での経験，○計算機分野での経験年数，○計算機分野での経験ステップ数，○当該プログラミング言語についての経験，○複雑な問題についての経験
環境にかかわる要素	仕様設計へのプログラマの参加，○プログラム論理の複雑さ，プログラム設計の複雑さ， ○プログラマの外国語の知識，共用ソフトウェ

	ア財産，　ドキュメンテイション・ページ数，参考文献の利用可能性，　他グループの協力，開発期間（納期），○参加スタッフ数，○開発ステップ数，　図形処理，○外部バイナリ・データ，○特殊入出力機能
計算機利用環境	バッチ処理の回転時間，○ＴＳＳ端末の利用可能性，○ＴＳＳ端末の応答の速さ，　メモリ容量の制約，○タイミングの制約，○障害発生
組織的開発体制	○主任プログラマ制，　構造化プログラミング，ウォークスルー，　関連分野についてのグループ学習，　トップダウン・デザイン，　開発ツール（ユーティリティ），　記録作成用ユーティリティ

上記３種の作業についての生産性指標と生産性の関係を図-2に示す．

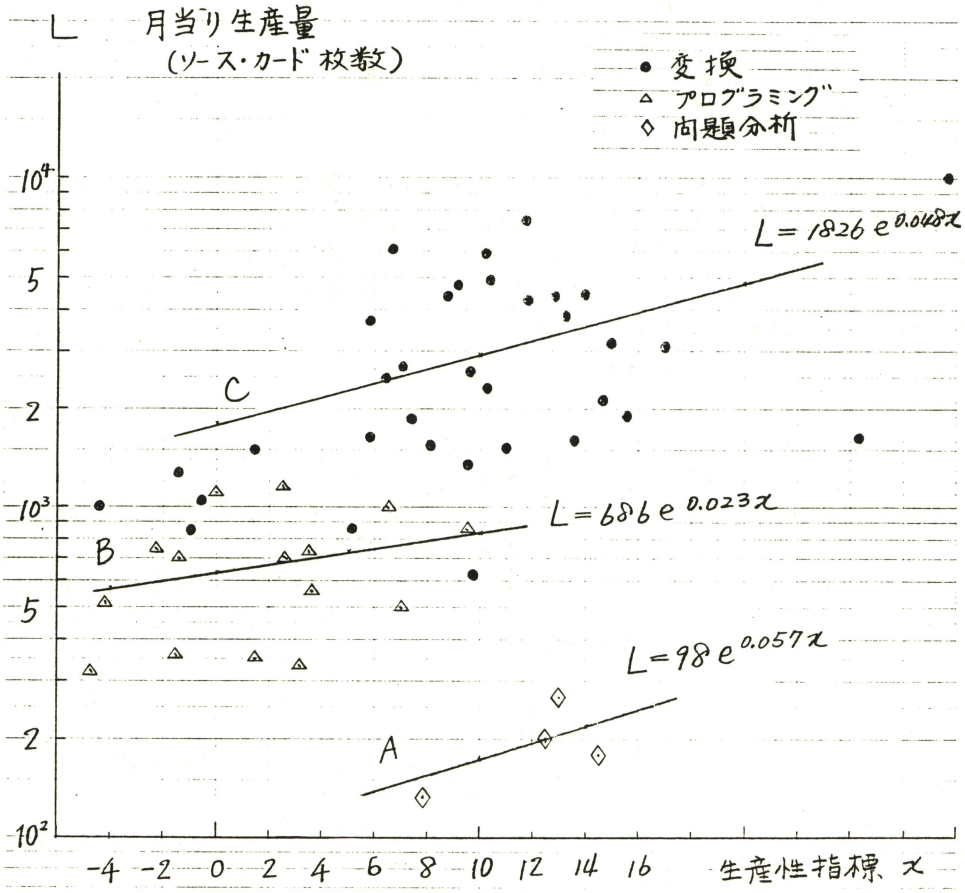

図-2　生産性指標と生産量

この図-2からわかるとおり，生産性指標が直ちに生産性を示すものではないことがわかる．しかし問題分析，改良，定型的プログラミング，単純変換などに局面を限定すれば生産性指標は生産性の目安になる．

3. Halsteadの Software Science 理論

Halsteadはプログラミング作業の生産性に関し，Software Scienceと呼ぶ理論を発表している．[3,4,5] この理論は生産されたプログラムから，そのプログラムを構成するオペレイタ（演算子）とオペランドの数を読み取り，その値をもとにそのプログラムを生産するに要したプログラマの時間を推定するものである．この理論は以下の計算式で示される．

n_1：ひとつのプログラム中に現われた異ったオペレイタの数．
n_2：ひとつのプログラム中に現われた異ったオペランドの数．
N_1：ひとつのプログラム中に現われたオペレイタの出現回数の総和．
N_2：ひとつのプログラム中に現われたオペランドの出現回数の総和．

vocabulary n： $n = n_1 + n_2$，　プログラム長： $N = N_1 + N_2$

推定プログラム長 $\hat{N}$： $\hat{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2$．

Volume V： $V = N \log_2 n$．

プログラム・レベル L： $L = (2/n_1)(n_2/N_2)$．

言語レベル λ： $\lambda = L^2 V$．

プログラム作成に要する努力（effort）E： $E = V/L$．

プログラム作成に要する時間 T： $T = E/S$．ここで S はプログラマが1秒間に認識できるプログラム要素の数を表わしている．Halsteadは心理学者Stroudの実験結果から S の値として18を採用している．

Halsteadは，Comm. ACM に発表されている20のアルゴリズムについて実験をおこない，この理論の妥当性を主張している[3]．また，彼は前節2.のWalston and Felixの報告についても疑問を投げかけ，同じデータを使って自己の理論の妥当性を主張した[6]．

この理論を図-2の6個のプログラム（107個のサブルーチン，6287枚の実行文）に適用した結果を表-3に示す．観測値と理論値が大きくずれていることがわかる．

表-3　Halsteadの理論値と観測値との対比（1）

プログラム番号	実行ステップ	観測値（人月）	Halsteadの理論値（時間）	プログラムの性格
1	1812	2.5	191	定型的プログラミング
2	531	1.0	27.5	〃
3	1235	4.0	80	〃
4	420	1.0	172	改良
5	180	1.0	4	問題分析
6	1167	2.0	42.5	定型的プログラミング
7	706	1.0	64	〃

注：1人月は筆者らの場合は170時間に相当する．

このずれは何に起因するのであろうか．以下にそれらの原因となるものについて考えてみる．

(1) Halstead の理論は Comm ACM のアルゴリズム欄に発表されている20のプログラムについては良く合う．それらのプログラムはいずれもプログラム・ステップ数が10～50程度の小さなものばかりである．Halsteadがその報告に例として挙げている小さなプログラムについては，筆者らの解析プログラムもHalsteadの結果と一致する．このような小さなプログラムではHalsteadが採用しているStroud 数Sの値S＝18は妥当のようである．このように小さなプログラムでは，プログラムを読んでその内容を理解する時間と，仕様書を読んでそれをプログラムに書き下す時間とは大差がなく，そのとき現われる記号を人間は1秒間に18個認識できるということになる．

(2) この考え方からするとHalsteadの理論がよく適合するのは，

計画，要求分析，設計，コーディング，テスト，保守

のように分類した作業のうち，設計＋コーディングのみのはずである．表-3のプログラムについて，この考え方にしたがって設計＋コーディングに要した時間のみを取り出して理論値と対応させたのが表-4である．表-4からわかるとおりこのような考え方でもうまく合わないようである．

表-4　Halsteadの理論値と観測値との対比（2）

プログラム番号	実行ステップ	観測値（人月）	Halsteadの理論値（時間）	プログラムの性格
1	1812	0.5	191	定型的プログラミング
2	531	0.3	27.5	〃
3	1235	0.5	80	〃
4	420	0.3	172	改良
5	180	0.4	4	問題分析
6	1167	0.9	42.5	定型的プログラミング
7	706	0.5	64	〃

(3) プログラム中に現われる要素の数を示す$\hat{N}$と観測値Nとの対比は図-3のようになる．この点まではHalsteadの理論は筆者らの観測値とも合っている．

(4) 言語レベルλはHalsteadによれば表-5のとおりである．筆者らの観測結果ではλは定数ではなく，図-4のとおりステップ数EOS，したがって$\hat{N}$の関数と見るのが妥当のようである．プログラムをライン・プリンタ1ページに収まる程度のサブルーチン群に分割すればλは1.0～1.6になるであろうが，その程度まで詳細にプログラムを分割するに要する時間はHalsteadの理論からは計算できない．

表-5
Mean and Variance of Language Level for Seven Languages

Language	λ	Variance
English	2.16	0.74
PL/I	1.53	0.92
Algol 58	1.21	0.74
Fortran	1.14	0.81
Pilot	0.92	0.43
Assembly	0.88	0.42

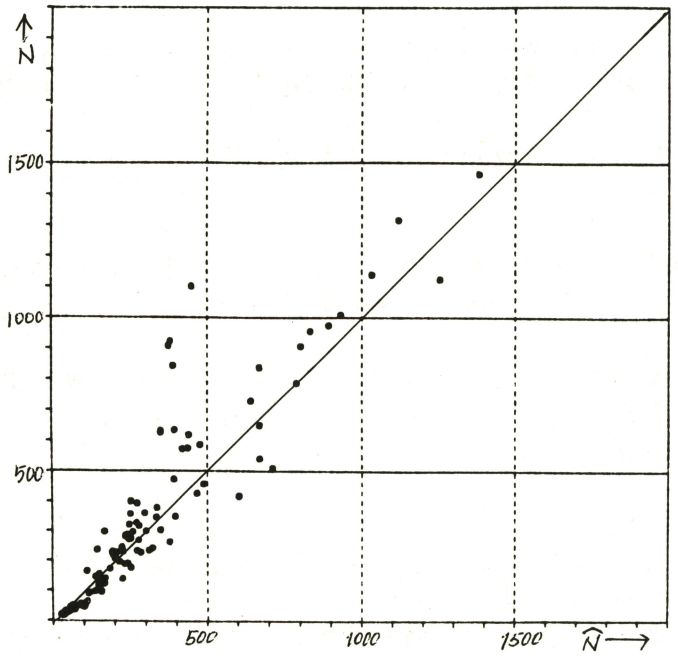

図-3　プログラム長の理論値($\hat{N}$)と観測値

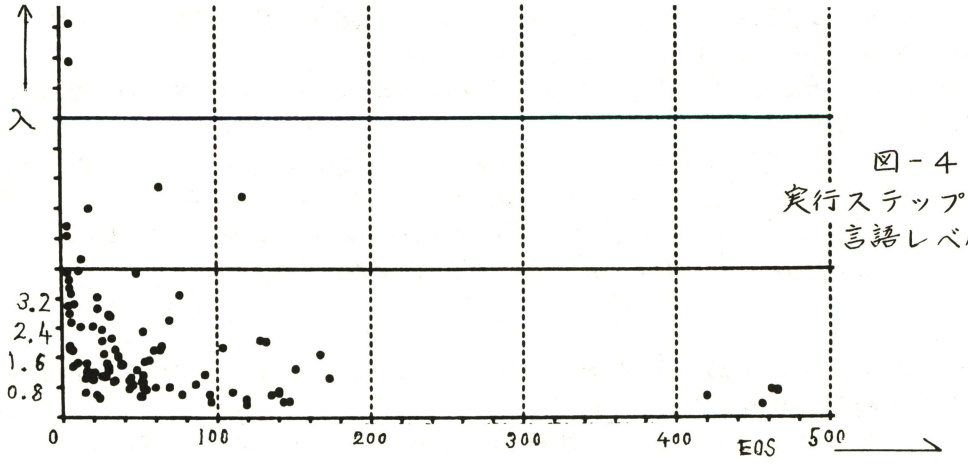

図-4
実行ステップ(EOS)と言語レベル(入)

(5) Halstead理論についてのもうひとつの疑問はStroud数Sの値を作成されるプログラムの大小にかかわりなく常に一定(S=18)としていることである。この値もやはり$\hat{N}$の関数とみるのが妥当ではないだろうか。その根拠として次の2つの仮設が考えられる。

(a) 人間は疲労する。数ヶ月間継続して毎秒18個のプログラム要素を認識することはできない。

(b) 人間の脳の作業領域(バッファ・メモリ)に1度に収めることができるプログラム要素数は比較的小さく、ステップ数にして30程度である。

(6) 以上Halsteadの方法は理論としては興味あるが、現実への適用可能性についてはさらに検討の必要がある。

4. 結論

Walston and Felix の方法と Halstead の理論はともに興味深いが今後さらに検討をかさねる必要がある．この段階で筆者らが科学技術計算のソフトウェア開発について言えることは以下のようなことである．

(1)計算機プログラムの開発，変換に要する費用を，ある程度の誤差の範囲内で，予測することが可能である．

(2)プログラムの開発，変換作業には，作業の難易度に応じて3つの基本的な階層レベルがあり，一般の作業はこの3レベルの組み合せで表現される．

(3)プログラムの開発，変換には基本となる3つの局面とこの3つの適当な組み合せで得られる局面の合計4つの局面が存在する．したがって局面を限定しないで漠然と生産性を議論することは間違いである．

参考文献

1) C.E. Walston and C.P. Felix, *A method of programming measurement and estimation*, IBM Systems Journal No.1, 1977.
2) プログラミングにおける生産性と開発工数の見積もり方，日経エレクトロニクス，1978年9月18日号．
3) M.H. Halstead, *Software physics: basic principles*, RJ 1582, IBM Research, Yorktown Heights, N.Y., 1975.
4) M.H. Halstead, *Elements of software science*, Elsevier North-Holland, Inc., N.Y., 1977.
5) A. Fitzsimmons and T. Love, *A Review and Evaluation of Software Science*, Computing Surveys, Vol. 10, No. 1 March 1978.
6) M.H. Halstead, *On lines of code and programming productivity*, IBM Systems Journal, (Forum), No. 4, 1977.

附録

問題分析はむづかしい問題を計算機プログラム化しようとするときの生産性を示す．研究開発機関に典型的な例である．

プログラム作成に使用すべき解法が物理的に正当であるかどうか，数値計算の方法が妥当であるかどうかがはっきりしていない問題を解こうとしているときである．

この問題について誤解を避けるために簡単な説明を加えておこう．

ここであらわれた4つの問題はすべて核融合の研究に関するものである．それらはつぎの2つに大別される；

(a)MHDプラズマ安定性を調べる際にあらわれる大次元の固有値問題を解き，所要の固有値を求めること，

(b)JT-60真空容器周りの自然対流と温度分布を求めること，である．

上記(a)の問題を解析している核融合理論解析研究室の研究者2名はともに工学博士の学位をもつ．(b)の問題を解析している大型トカマク部の研究者は工学の分

野で博士の学位をもつ．また，これらの問題の理論式を数値解析の問題に変換し，プログラム作成をおこなった計算センタの外来研究員（富士通派遣）も物理の分野で博士の学位をもち，計算機のプログラム作成に関し，約6年の経験をもつ．

まず(a)の問題について簡単にながめてみよう．

MHD プラズマの安定性を調べる際にあらわれる大次元の固有値問題はすでに確立した手法がある．理論解析研究室でも現在はその手法によって問題を解いているが，この手法の難点は計算機のメモリと計算時間をくうことである．そのために現在でもジョブの回転時間はよくない．一年後には計算時間8倍，メモリ4倍の問題を大量に解きたいが，現在の研究所の計算機の能力からみて，この希望の実現はむづかしい．

そこで固有値問題の新しい解法の採用が検討されることになった．文献が調査され，不完全ランチョス法が試みられた．

この手法の適用可能については，日本において非常に高名な2人の研究者が，それぞれ独立に，実例をあげて肯定的な解説をおこなっている．

しかし採用された手法では所要の固有値を得ることができなかった．

この手法は性質の良い行列（データ）でしかその適用可能性が確かめられていないのであるが，MHD 安定性解析にあらわれる行列は極端に性質が悪く，必要な固有値が求められなかった．こうして当初の期待ははずれてしまったのであるが，この問題を解析する過程で，出題者とプログラム分析・作成担当者は新しい手法を思いついた．この手法は近日中に実験される予定であるが，正しい解を出すものと関係者はその結果に希望をもっている．

これがうまくゆけば，その結果は論文となってあらわれ，今後この問題を解く人はこれを直線B（定型化されたプログラミング）の問題としてあつかうことができることになる．

つぎに(b)の問題について簡単にみてみよう．

これはJT60真空容器の周囲に生じる自然対流と温度分布を解析しようとするものである．物理的には Navier－Stokes の流体方程式と熱伝達の方程式を解くことになる．このとき(i)時間依存でない定常状態，(ii)簡単な幾何学的形状という2つの条件のもとでは，この問題を解いた先例がある．

この問題を解くにあたって出題者から出された条件は，(iii)複雑な形状，(iv)複雑な座標系のとり方，などであった．

この問題を数値計算によって計算機で計算したところ，解をうまく求めることができなかった．もともとこの種の問題を解くことはむづかしいことがわかっているが，今回の原因は複雑な座標の特点となる部分で，数値的な誤差が入りこむことにあるのではないかとプログラム作成者は予測している．座標系を簡単にするなどによって正しい解が求められる可能性があり，今後はこの方向で作業を継続する考えである．この作業に1ヶ月かければ図-2の直線Aは点線A'となる．

このような問題分析に関するプログラムの作成は通常は研究者側の作業として処理され表面にはでてこない場合がほとんどである．うまくいった場合には論文その他の成果となってあらわれる．