プログラムの変更に対するスライス計算の適応度*

6 L − 4

根岸 純，太田 剛，水野 忠則†
静岡大学大学院理工学研究科‡

1 はじめに

プログラムスライスとは，プログラム P 中の実行地点 n と変数 v が与えられたとき，n における v の値に影響を及ぼす可能性のある P 中の文や式の集合のことを言う [4]．当初はプログラムのデバッグを支援するために考えられていたものであるが，今日ではソフトウェアのテスト，デバッグ，調査，及び保守に効果的であることが認識されている．

一般にデバッグや保守作業においては，プログラムテキストに変更を加える．しかし，これまでに提案されたスライス計算アルゴリズムに対し，テキスト変更を想定した場合の十分な評価は行われていない．そこで本稿では，これらのアルゴリズムを実装し，プログラムテキストに変更が生じた場合の再計算量に関する実測実験を行い，その適応度を調査する．

2 既存のアルゴリズム

現在提案されているスライス計算アルゴリズムは，4種類存在する： （1） **Weiser** は，手続き呼出を含まないプログラムのスライスを計算するために，プログラムの処理の流れをフローグラフで表し，値の定義・参照関係についてデータフロー方程式を立て，これを解くことによってスライスを求めることを示した [4]． （2） Ottenstein は，プログラムテキストを解析してプログラム依存グラフ（Program Dependence Graph 以下 PDG）を構成することによって，スライス計算を単純なグラフ探索問題に置き換えられることを示した [3]． （3） Bergeretti は，一度プログラムテキストから μ 関係行列を計算することにより，スライスを線形時間で求めることが可能であることを示した [1]． （4） また，太田はプログラムテキストを解析して変数依存グラフ（Variable Dependence Graph 以下 VDG）を構成することによって，PDG と同様にスライス計算をグラフ探索問題に置き換えられることを示した [2]．

3 実験計画と評価

モデル言語

本研究で扱うモデル言語を説明する．これは手続き型言語であり，構造型プログラミング言語に最低限必要な文をもつ．すなわち代入文，if 文，if-else 文，while 文，入力文，出力文，変数宣言から構成される．変数はスカラ型に限る．配列，構造体，ポインタ等はない．

実験計画

プログラムスライスは，変数の値の受渡しによる依存関係や，制御文による依存関係を解析することで計算される．各アルゴリズムは，プログラムテキストを解析することでこの関係を固有の中間表現に置き換え，これを用いてスライス計算を行う（図 1）．したがって，中間表現再構成の計算量を抑えることができるアルゴリズムは，プログラム変更に対する適応度が優れているといえる．そこで中間表現の計算時間及びその再計算時間を実測する．

表 1は各アルゴリズムの計算量の理論値を表している [2]．ただし，文の数を S，変数の数を V とする．なお，Weiser のアルゴリズムにおける平均的なプログラムに対する計算量は不明なので，表には記入していない．

この表から，それぞれのアルゴリズムは，実行文数や変数の数に対する計算量の依存度が異なることがわかる．したがってこれらの特徴を正確に把握するには，対象とするプログラムの種類がある程度必要であると考えられる：そこで，（1）同じ実行文数で変数の個数が数個から数十個の幅を持つプログラム群，（2）変数の個数は同じで実行文数が数行から 300 行程度まで幅のあるプログラム群の 2 種類のプログラム群を用意し測定を行う．そして用意したプログラムのそれぞれに対し，文の追加や削除といった変更情報を与え，一旦生成した中間表現

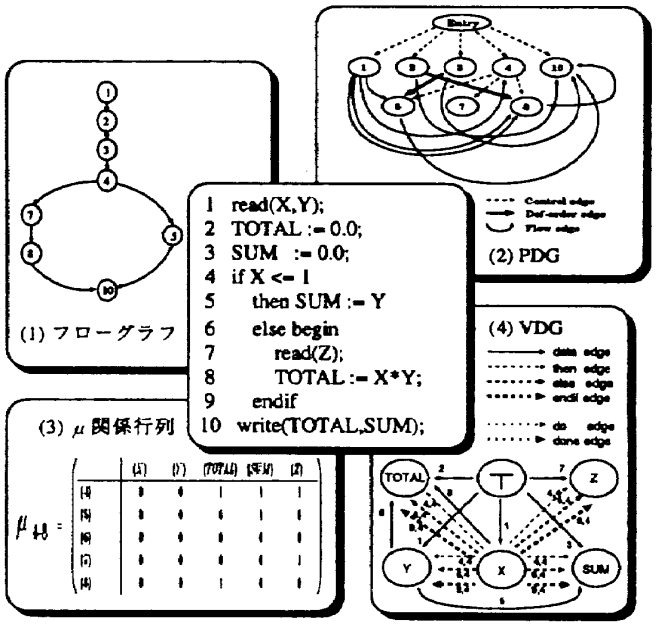

図 1: 各アルゴリズムが用いる中間表現

*A Suitability of Slicing Algorithms for Program Modification
†Jun Negishi, Tsuyoshi Ohta and Tadanori Mizuno
‡Graduate School of Science and Engineering, Shizuoka University.

提案者 中間表現	テキストから中間表現		中間表現からスライス		プログラム変更	
	最悪の場合	平均	最悪の場合	平均	最悪の場合	平均
Weiser フローグラフ	$O(S(S+V))$	—	$O(S^3 \log S)$	—	$O(S+V)$	—
Bergeretti PDG	$O(S^2V)$	$O(S \log S)$	$O(S^2V)$	$O(S \log S)$	$O(S^2V)$	$O(S \log S)$
Ottenstein μ 関係行列	$O(V^2(S+V))$	$O(V^2S)$	$O(S)$	$O(S)$	$O(V^2(S+V))$	$O(V^2S)$
太田 VDG	$O(S^2V)$	$O(S \log S+V)$	$O(S^2V^2)$	$O(SV \log S)$	$O(SV)$	$O(S)$

表 1: スライス計算アルゴリズムにおける計算量の比較

の再計算を行う．

今回はプログラム文の数S {30,80,130,180,230,280}，変数の数V {2,3,4,5,6,7,8,9,10,11,12} の幅で測定を行う．ここでは，新たな機能をプログラムに追加するしたり，他の計算機へプログラムを移植したりする場面を想定する．すなわち，個々のプログラムに施される変更はプログラム中にまんべんなく行われ，その規模は全体の文の数の約5% とする．

実験評価

テキストから中間表現を構成する時間を図2に，また中間表現の再構成にかかる時間を図3にまとめる．ただしこれはSと時間との関係を示したもので，$V = 6$のときのものである．

これらの図からわかるように，VDG を用いるアルゴリズムは中間表現の再構成時間を抑えていることがよみとれる．したがって，VDG を用いるアルゴリズムは，プログラムの変更に対するスライス計算の適応度は高いと考えられる．

4　おわりに

本稿では，プログラムの変更に対するスライス計算の適応度を調査するため，既存のアルゴリズムの時間計算量を実測し，その結果の分析を行った．

今後は，デバッグ作業のように，変更が一部に集中する状況や変更の規模の肥大等，様々な状況を想定して調査する予定である．このような場面では，プログラムテキストの変更回数がスライス計算回数をはるかに上回ると予想されるので，VDG によるアルゴリズムはさらに有利になると予想される．

参考文献

[1] Jean-Francois Bergertti and Bernard A. Carré. Informaion-frow and data-frow analysis of while-programs. *ACM Transactions on Programing Languages and Systems*, Vol. 7, No. 1, pp. 37–61, jan 1985.

[2] Tuyoshi Ohta, Takashi Watanabe, and Tomonori Mizuno. A slicing algolithm suitable for program modification. *IEICE Tramsaction on Fundamentals*, Vol. **E78–A**, No. 4, 1996.

[3] Ottenstein K.J. and Ottenstein L.M. The programing dependence graph in a software depelopement environment. *ACM SIGPLAN Notice*, Vol. 95, No. 5, pp. 177–184, 1984.

[4] Mark Weiser. Program slicing. *IEEE Transacions of software engineering*, Vol. **SE–10**, No. 4, pp. 352–357, jul 1984.

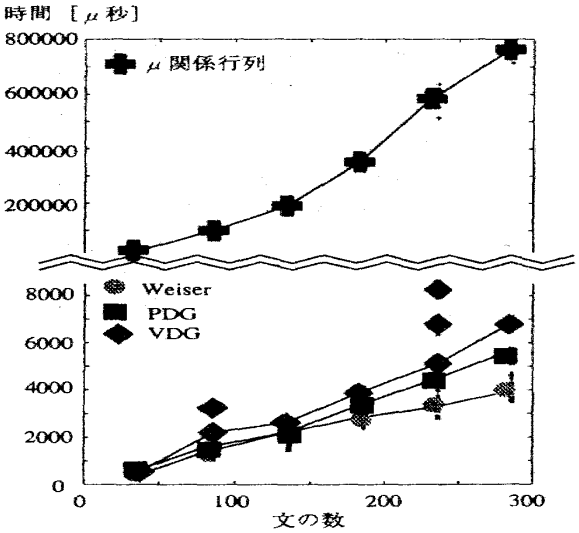

図 2: テキストから中間表現（変数の数=6）

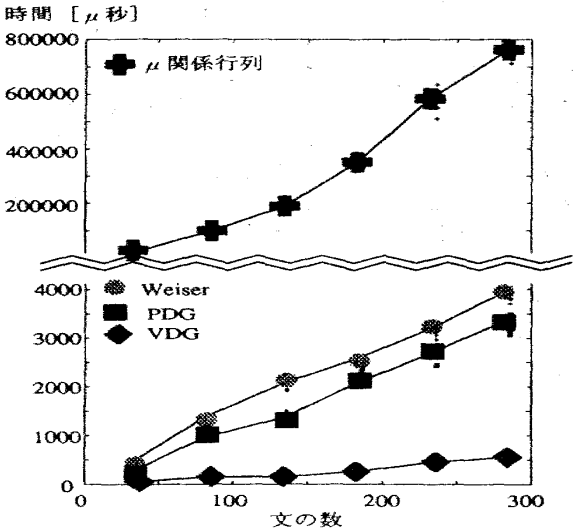

図 3: 中間表現の再構成（変数の数=6）