整数計画問題としてのループ並列実行時間の下限算出問題

中西 恒夫†, 城 和貴†, Constantine D. Polychronopoulos‡,
福田 晃†, 荒木 啓二郎†
†: 奈良先端科学技術大学院大学 情報科学研究科
630-01 奈良県生駒市高山町 8916 番地の 5
‡: Center for Supercomputing Research and Development,
University of Illinois at Urbana-Champaign
1308 West Main St. Urbana, IL 61801, USA
E-mail: nakasu-para@is.aist-nara.ac.jp

概要

並列処理の分野において、プログラムの並列実行時間の見積りは、静的スケジューリング、自動並列化等の応用で重要な問題である。特に、プログラム中のループには多くの並列実行可能部分が潜在するため、ループ部の並列実行時間の見積りに対する需要は大きい。ループ全体の並列実行時間の下限は、ループを展開したコードの依存グラフを生成し、そのクリティカルパスのコストを算出することにより得ることができるが、この方法はループの大きさに対するスケーラビリティの点で劣る。本稿では、ループを展開することなく、ループを展開したコードの依存グラフのクリティカルパスのコスト—すなわちループ全体の並列実行時間の下限—を算出する問題を、整数計画問題に帰着させ、単体法ならびに分枝限定法を用いて解く。これにより、問題解決にかかる時間はループの大きさに依存せず、大規模なループの並列実行時間の下限の算出が可能となる。

Evaluating an Infimum of Parallel Execution Time of the Loop

Tsuneo Nakanishi†, Kazuki Joe†, Constantine D. Polychronopoulos‡,
Akira Fukuda†, Keijiro Araki†
†: Graduate School of Information Science,
Nara Instituite of Science and Technology
8916-5, Takayama-cho, Ikoma-shi, Nara 630-01, Japan
‡:Center for Supercomputing Research and Development,
University of Illinois at Urbana-Champaign
1308 West Main St. Urbana, IL 61801, USA
E-mail: nakasu-para@is.aist-nara.ac.jp

ABSTRACT

It is an important problem to estimate parallel execution time of a given program in parallel processing. Especially estimating parallel execution time of loops is in demand since loops contain rich parallelism. The infimum of parallel execution time of a whole loop can be obtained by unrolling a given loop, generating a dependence graph of the unrolled loop, and evaluating a critical path cost of the dependence graph. Obviously this method lacks scalability against the loop size. In this paper we let it be an integer programming problem to evaluate a critical path cost of the dependence graph for the unrolled code of a given loop without unrolling the loop and solve it with the simplex method and a branch-and-bound algorithm. As a result time for solving the problem becomes independent of the loop size, thus we can evaluate infimums of parallel execution time of large loops.

1 はじめに

マルチプロセッサシステム上でのスケジューリングの問題は、しばしば有向グラフによってモデル化される。このグラフは依存グラフと呼ばれ、節点は個々のタスク、枝 (u, v) は節点 v のタスクが節点 u のタスクの実行完了を待たずに実行を開始できないことを意味する。依存グラフの節点には、該当するタスクの実行コスト（実行時間）が属性として付与される。また、依存グラフの枝 (u, v) には、節点 u のタスクから節点 v のタスクへ制御を移行する時のオーバーヘッド、例えば通信オーバーヘッドや同期オーバーヘッドが属性として付与される。この時、依存グラフのクリティカルパス—すなわちパスを構成する節点、枝のコストの総計が最大となるパス—のコストは、プロセッサ無限個時の並列実行時間の下限を与える。

逐次プログラムから、それと等価なマルチプロセッサシステム用並列プログラムを生成する自動並列化コンパイラでは、しばしば依存グラフがプログラムの中間表現として用いられ、種々の並列化処理に活用される [3]。依存グラフの節点は逐次プログラムのひとつの文に、枝 (u, v) は節点 u の文が実行されない限り節点 v の文を実行できないことを意味する。こうした依存関係は、データの生産者消費者関係に起因するもの（データ依存）や条件分岐に起因するもの（制御依存）がある [1]。本稿ではデータ依存のみを考えることにする。データ依存に関する枝については、データの通信コストを枝の属性として付けることができる。自動並列化コンパイラは、これらの実行コスト、通信コストを用いて、プログラムの並列実行時間の見積りを定量的に行う。

ループには並列実行可能部分が多く潜在するため、ループの並列実行時間の見積りは極めて重要である。プログラムのループ部分では、ループ内の各文は繰り返しの数だけ実体が存在するため、依存グラフは循環グラフになり得る。この時はクリティカルパスを定義することはできない。ループを展開すれば通常の非ループのコードが得られ、その依存グラフ上にクリティカルパスを定義することが可能となる。そのコストはループ全体をプロセッサ無限個で並列実行した際の実行時間の下限を与える。この方法は従来のクリティカルパス算出アルゴリズムが使用できる反面、ループの大きさに対するスケーラビリティは極めて低く、実用上明らかに問題がある。

一般にループには規則性があり、ループの個々のイタレーション（繰り返し）間で、ループ内部の文の実行コストや文の間のデータ通信のコストに差がないことが多い。本研究ではこの点に着目し、ループを展開することなくループを展開したコードの依存グラフのクリティカルパスのコスト、すなわちプロセッサ無限個時のループの並列実行時間の下限を算出する問題を、整数計画問題に帰着させる。2 章ではループにおける依存グラフについて諸概念の整理を行う。3 章では問題をいかに整数計画問題に帰着させるかを説く。

2 ループにおける依存グラフ

本節では、ループにおける依存グラフに関する諸概念の整理を行う。依存グラフでは、タスクと節点、依存と枝が一対一の対応となるため、本稿では両者を適宜入れ換えて用いることがある。

2.1 ループタスクグラフ

ループボディ内のタスク間の依存関係を表現した依存グラフを、**ループタスクグラフ**と呼ぶことにする。図 1(a) のプログラムのループタスクグラフは、図 1(b) のようになる。この例のように、一般にループタスクグラフは循環であり、クリティカルパスは定義できない。

ループタスクグラフの依存枝は、次の 2 種類に分類される [1]。

- **ループ独立依存 (loop-independent dependence)**—依存元タスクと依存先タスクがループの同じイタレーションの中にある依存。
- **ループによって運ばれる依存 (loop-carried dependence)**—依存元タスクと依存先タスクがループの異なるイタレーションにある依存。

ループによって運ばれる依存について、依存元タスクの n 個先のイタレーションに依存先タスクがある場合、その依存の**依存距離**は n であるという。ループ独立依存の依存距離は 0 である。ループタスクグラフの自己ループは、常にループによって運ばれる依存の枝となる。図 1(b) の点線の枝はループによって運ばれる依存、その脇の数字は依存距離を表している。

2.2 展開されたループタスクグラフ

ループを展開したコードの依存グラフは、基本ブロックの依存グラフと同様であり、非循環グラフとなる。図 2は、この依存グラフとループタスクグラフ、イタレーショングラフの関係を図解したものである。

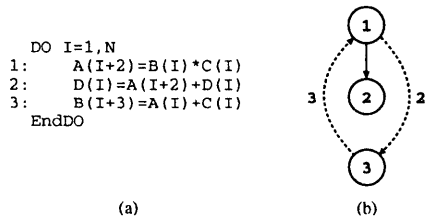

図 1: ループにおける依存グラフ

図2では、ループを展開したコードの依存グラフを、各イタレーションをひとつの層として、立体的に描いている。異なるイタレーションの同じ文に相当する節点は、それぞれの層で同じ位置に配置している。各層に現れるグラフは、ループタスクグラフから、ループによって運ばれる依存の枝を取り除いたものとなる。また、層と層の間にはループによって運ばれる依存の枝があり、その枝の始点と終点の高度差が依存距離に相当する。このグラフを**展開されたループタスクグラフ**と呼ぶことにする。

展開されたループタスクグラフの真上から平行光線をあてた時に、展開されたループタスクグラフの真下に結ぶ像が、ループタスクグラフとなる。本稿では、ループタスクグラフの節点 v に相当する、ループ展開後の節点 vv_1, vv_2, ..., vv_N を、v の投影元の節点[1]と呼ぶことにする。逆に、節点 v を節点 vv_1, vv_2, ..., vv_N の投影先の節点と呼ぶことにする。また、枝、パスに対しても同様の表現を用いる。

展開されたループタスクグラフのクリティカルパスのコストは、ループをプロセッサ無限個で並列実行した際の実行時間の下限を与える。

2.3 パスの影

図2は、展開されたループタスクグラフにおけるパス（図中太線）が、ループタスクグラフ上にどのように投影されるかを示している。この例のように、ループタスクグラフ上に投影された、展開されたループタスクグラフ上のパス p の像を、パス p の影と呼ぶことにする。

ループタスクグラフの枝にパスの影が何回重なるかを、その枝の**影重複度**と呼ぶことにする。例えば、図2のループタスクグラフの場合、枝 a, b, c の影重複度はそれぞれ 2, 0, 1 である。

[1] 一般的にはタスク u のインスタンスと呼ばれる。

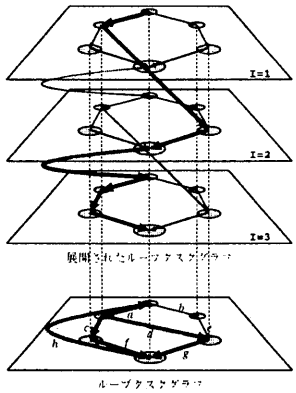

図 2: 展開されたループタスクグラフ

2.4 ループタスクグラフの前処理

問題の取り扱いを容易にするため、ループタスクグラフに次の前処理を加えるものとする。

まず、2つの仮想的な節点 *START*, *STOP* を設ける。両者のコストはいずれも0とする。*START* から、ループタスクグラフのループによって運ばれる依存を削除した時に極小となる節点[2]に向けて、コスト0、依存距離0の枝を張る。また、ループタスクグラフのループによって運ばれる依存を削除した時に極大となる節点[3]から、*STOP* に向けて、コスト0、依存距離0の枝を張る。後に証明するが、この処理により、展開されたループタスクグラフにおけるクリティカルパスは、*START* の投影元節点のひとつを始点とし、*STOP* の投影元節点のひとつを終点とすることが保証され、問題の取り扱いが容易になる。

3 整数計画問題への帰着

ループ全体のプロセッサ無限個による並列実行時間の下限は、展開されたループタスクグラフのクリティカルパスのコストにより与えられる。本節では、展開されたループタスクグラフのクリティカルパスのコスト算出の問題が、いかに整数計画問題へ帰着されるかについて述べる。

なお本稿では、記述を正確かつ簡潔にすべく、形式仕様記述言語Zの表記法を部分的に用いている。Zの

[2] 入る枝のない節点は極小であるという。
[3] 出る枝のない節点は極大であるという。

詳細については文献[2]を参照されたい。また、付録Aに本稿で用いたZの記号をまとめている。

3.1 表記と定義

本稿では、与えられたループタスクグラフを $G=(V,E)$、Gに対応する展開されたループタスクグラフを$\tilde{G}=(\tilde{V},\tilde{E})$と表記する。$V$, Eはそれぞれループタスクグラフの節点、枝の集合、$\tilde{V}$, $\tilde{E}$はそれぞれ展開されたループタスクグラフの節点、枝の集合である。その他、本稿で用いる記号の表記と定義を付録Bにまとめる。

本稿では、展開されたループタスクグラフ上のパスを、展開されたループタスクグラフの節点の列をもって定義する。また、パスのコストを、パスを構成する節点と枝のコストの総計と定義する。すなわち、パス $p=\langle vv_1, vv_2, ..., vv_n\rangle$ のコスト$\Omega(p)$は、形式的に式(1)のように定義される。

$$\Omega(p)=\omega_{\tilde{V}}(vv_1)+\sum_{i=1}^{n-1}(\omega_{\tilde{E}}((vv_i,vv_{i+1}))+\omega_{\tilde{V}}(vv_{i+1})) \quad (1)$$

3.2 対象ループ

仮定 1 本稿で対象とするループは次の条件を満足するものとする。

1. $\forall vv:\tilde{V}\bullet\omega_{\tilde{V}}(vv)=\omega_V(f_{\tilde{V}}(vv))$
2. $\forall ee:\tilde{E}\bullet\omega_{\tilde{E}}(ee)=\omega_E(f_{\tilde{E}}(ee))$
3. $\forall ee:\tilde{E}\bullet d_{\tilde{E}}(ee)=d_E(f_{\tilde{E}}(ee))$
4. ループの入れ子は一段。
5. ループの繰り返し数は定数N。
6. ループボディ内に条件分岐がない。□

1-3は、展開されたループタスクグラフの任意の節点のコスト、枝のコストおよび依存距離は、ループタスクグラフ上の投影先の節点のコスト、枝のコストおよび依存距離に等しいことを意味している。

3.3 パスの影の性質

クリティカルパスの影には次の性質がある。

性質 1 展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$のクリティカルパスは、ループタスクグラフ$G=(V,E)$の$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とする。(証明略) □

本稿で問題としているのはクリティカルパスのコストの算出である。性質1により、我々は$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とするパスのみを考慮の対象とすればよいことになる。

一方、仮定1の下では、一般のパスの影には以下の性質がある。

性質 2 展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$上に、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とする、パス$p$が与えられている。この時、枝$e$の測度$\varphi_E(e)$として枝$e$の影重複度を充てると、パス$p$のコスト$\Omega(p)$は式(2)により算出することができる。(証明略) □

$$\Omega(p)=\sum_{(u,v)\in E}(\omega_E((u,v))+\omega_V(v))\varphi_E((u,v)) \quad (2)$$

性質 3 展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$上に、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とする、パス$p$が与えられている。この時、枝$e$の測度$\varphi_E(e)$として枝$e$の影重複度を充てると、次の関係式(3)が成り立つ。(証明略) □

$$\sum_{e\in E}d_E(e)\varphi_E(e)<N \quad (3)$$

性質 4 展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$上に、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とする、パス$p$が与えられている。この時、枝$e$の測度$\varphi_E(e)$として枝$e$の影重複度を充てると、ループタスクグラフ$G=(V,E)$の節点$START$, $STOP$について、式(4)が成り立つ。(証明略) □

$$\sum_{(START,v)\in E}\varphi_E((START,v)) = \sum_{(v,STOP)\in E}\varphi_E((v,STOP))=1 \quad (4)$$

性質 5 展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$上に、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とするパス$p$が与えられている。この時、枝$e$の測度$\varphi_E(e)$として枝$e$の影重複度を充てると、ループタスクグラフ$G=(V,E)$の、$START$, $STOP$を除く任意の節点$v\in V\setminus\{START,STOP\}$について、式(5)が成り立つ。(証明略) □

$$\sum_{(u,v)\in E} \varphi_E((u,v)) - \sum_{(v,u)\in E} \varphi_E((v,u)) = 0 \quad (5)$$

3.4 整数計画問題への帰着

ループタスクグラフ $G=(V,E)$ の任意の枝 $e \in E$ には、測度$\varphi_E(e)$ が定義される。本稿では、ループタスクグラフの全ての枝の測度への値の割り当ての方法を、**測度定義**と呼ぶことにする。

前小節では、展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$ 上に、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とするパスを描き、ループタスクグラフ $G=(V,E)$ の任意の枝 $e \in E$の測度$\varphi_E(e)$ として、その時の枝 eの影重複度を割り当てた場合、その測度定義は式 (3)–(5) を満足することを述べた。本小節では逆に、最初に式 (3)–(5) を満足するような測度定義が与えられた時、枝 eの影重複度が測度$\varphi_E(e)$ と等しくなるようなパスを、展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$ 上に描くことができるかどうかを考察してみる。

ここでループタスクグラフ $G=(V,E)$ より、測度0の枝を除いた後に孤立節点を除き、さらに残っている任意の枝 (u,v) に平行な $(\varphi_E((u,v))-1)$ 本の枝 (u,v) を追加して得られるグラフ G'を考える。グラフ G'は次の性質を有する。

性質 6 グラフ G'の連結成分のうちひとつは必ず節点 $START$, $STOP$の両方を含む。(証明略) □

$START$, $STOP$を含む連結成分について考える。式 (5) より、グラフ G'の $START$, $STOP$以外の節点の入次数と出次数は等しい。また式 (4) より、グラフ G'の $START$の出次数 (1) は入次数 (0) より 1 大きく、$STOP$の入次数 (1) は出次数 (0) より 1 大きい。したがって、$START$, $STOP$を含む連結成分には$START$より$STOP$に至るEuler路が存在する [4]。もし、グラフ G'が $START$, $STOP$を含む連結成分のみでなる場合、次のアルゴリズム 1により、式 (3)–(5) を満足するような測度定義より、ループタスクグラフ $G=(V,E)$ の枝 $e \in E$の影重複度が測度$\varphi_E(e)$ と等しくなるようなパス pを、展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$ 上に確定できる。このパス pは、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とする。

アルゴリズム 1

1. $i := 1$
2. $\widetilde{cur} := F_V(START)$ のうち最もイタレーション番号の若いもの
3. $p := \langle \widetilde{cur} \rangle$
4. $cur :=$ Euler路の i番目の節点
5. $cur = STOP$ならば、アルゴリズム終了。
6. $v :=$ Euler 路の $i+1$ 番目の節点, $vv :=$ $(\widetilde{cur}, vv) \in F_E((cur, v))$ なる$\tilde{G}$上の節点
7. $p := p \frown \langle vv \rangle$
8. $cur := v$, $\widetilde{cur} := vv$, $i := i+1$
9. 4へ戻る。□

先に式 (3)–(5) を満足するような測度定義を与えた時、ループタスクグラフ $G=(V,E)$ の任意の枝 $e \in E$の影重複度が測度$\varphi_E(e)$ と等しくなるようなパスは、一般に複数存在する。しかしながら、それらのパスのコストは、式 (2) により全て等しいことが保証される。

ここで、展開されたループタスクグラフ$\tilde{G}=(\tilde{V},\tilde{E})$ 上の、$START$の投影元節点のひとつを始点とし、$STOP$の投影元節点のひとつを終点とするパスにより与えられる影重複度を測度とする時の、測度定義の集合を A とする。また、式 (3)–(5) を満足する測度定義の集合を Bとする。性質 3–5により $A \subseteq B$である。よって、クリティカルパスのコストを与える測度定義は、Bの中から発見できるはずである。

以上の考察で、クリティカルパスのコストを求める問題は、式 (3)–(5) の規定する制約条件を満足する測度定義のうち、式 (2) の値を最大にし、かつグラフ G'が連結となるような (すなわち連結成分が唯ひとつとなるような) ものを探索する問題に帰着できる。全ての枝の測度の値は整数、式 (2) ならびに式 (3)–(5) は線形式であるので、問題は整数計画問題となる。

以下に整数計画問題の変数、目的関数、制約条件をまとめる。

変数 測度$\varphi_E(e)$ $(\forall e \in E)$

目的関数 式 (2)

制約条件 式 (3), (4), (5)

3.5 アルゴリズムの詳細

前小節の整数計画問題を解くにあたり、本研究では単体法と分枝限定法を用いる。単体法で得られる解は一般に実数であり、単体法で得られた実数解近傍に存在する整数解を探索するために、本研究では分枝限定法を用いる。また、式(3)-(5)を満足する測度定義のうち、パスを与えないもの（すなわちグラフ G'が非連結となるもの）を除くためにも分枝限定法を用いる。分枝限定法のアルゴリズムをアルゴリズム2に示す。単体法については、線形計画法に関する適当な文献を参照されたい。

分枝限定法では暫定解が定義されるが、本稿では暫定解を三つ組 (C, Ω, φ) と定義する。但し、Cは制約条件の集合、Ω, φ はそれぞれ、単体法によって得られた制約条件 Cの下での目的関数（式(2)）の最大値とその時の測度定義である。

アルゴリズム 2

1. 初期解 $(C_0, \Omega_0, \varphi_0)$ をリスト lisに挿入。但し、C_0 は式(3)-(5)のみからなる制約条件の集合である。
2. リスト lisよりΩ最大の暫定解 $(C_{cur}, \Omega_{cur}, \varphi_{cur})$ を取り出す。
3. 測度定義φ_{cur}の全ての測度が整数、かつ測度定義 φ_{cur}の与えるグラフ G'が連結であるならば、アルゴリズム終了。
4. 測度定義φ_{cur}より整数でない測度 valを持つ枝 e を選択。
5. $C_1 := C_{cur} \cup \{$ 制約条件$\varphi_E(e) \geq \lceil val \rceil\}$
6. 暫定解 $(C_1, \Omega_1, \varphi_1)$ をリスト lisに挿入。
7. $C_2 := C_{cur} \cup \{$ 制約条件$\varphi_E(e) \leq \lfloor val \rfloor\}$
8. 暫定解 $(C_2, \Omega_2, \varphi_2)$ をリスト lisに挿入。
9. 2へ戻る。□

4 まとめ

本稿では、ループを展開することなく、ループを展開したコードの依存グラフのクリティカルパスのコスト、すなわちプロセッサ無限個時のループの並列実行時間の下限を算出する問題を、整数計画問題に帰着させる方法について述べた。またこの問題を単体法と分枝限定法を用いて解くアルゴリズムを提案した。

クリティカルパスのコストは、コンパイル中に頻繁に計算されるケースが多い。本問題に特化した、整数計画問題を高速に解く専用アルゴリズムの開発が、今後の課題として挙げられる。

参考文献

[1] Hans Zima and Barbara Chapman, *Supercompilers for Parallel and Vector Computers*, Addison Wesley, 1990.

[2] J. M. Spivey, *The Z Notation A Reference Manual*, Prentice Hall, 1992.

[3] David F. Bacon, Susan L. Graham, Oliver J. Sharp, "Compiler Transformations for High-Performance Computing", *ACM computing surveys*, Vol. 26, No. 4, pp.345–420, December 1994.

[4] 守屋 悦朗, 『コンピュータサイエンスのための離散数学』, サイエンス社, 1992.

付録 A 本稿で用いた Z の記号

$\mathbb{N}$	0以上の整数
$\{a, b, c\}$	集合の定義
$\{v : type \bullet predicate\}$	集合の一般的な定義
$\mathbb{P}\, S$	集合 Sの巾集合
$a \mathinner{..} b$	a以上 b以下の整数の集合
$\#S$	集合 Sの要素数
$S \setminus T$	集合 Sと集合 Tの差
$\mathrm{seq}\, S$	集合 Sの元を要素とする列
$\langle a, b, c \rangle$	列
$s_1 \frown s_2$	列の連結
$X \rightarrow Y$	Xから Yへの全域関数
$X \nrightarrow Y$	Xから Yへの部分関数
$X \twoheadrightarrow Y$	Xから Yへの全域全射

付録 B 記号の表記と定義

表記	定義（型）
N	ループの繰り返し数（$\mathbb{N}$）
$\Omega(p)$	$\widetilde{G}$におけるパス pのコスト（$\mathrm{seq}\, \widetilde{V} \nrightarrow \mathbb{N}$）
$\omega_V(v)$	Gの節点 vのコスト（$V \rightarrow \mathbb{N}$）
$\omega_E(e)$	Gの枝 eのコスト（$E \rightarrow \mathbb{N}$）
$d_E(e)$	Gの枝 eの依存距離（$E \rightarrow 0 \mathinner{..} N-1$）
$\varphi_E(e)$	Gの枝 eの測度（$E \rightarrow 0 \mathinner{..} N$）
$f_{\widetilde{V}}(vv)$	$\widetilde{G}$の節点 vvの Gにおける投影先の節点を返す関数（$\widetilde{V} \twoheadrightarrow V$）
$f_{\widetilde{E}}(ee)$	$\widetilde{G}$の枝 eeの Gにおける投影先の枝を返す関数（$\widetilde{E} \twoheadrightarrow E$）
$\omega_{\widetilde{V}}(vv)$	$\widetilde{G}$の節点 vvのコスト（$\widetilde{V} \rightarrow \mathbb{N}$）
$\omega_{\widetilde{E}}(ee)$	$\widetilde{G}$の枝 eeのコスト（$\widetilde{E} \rightarrow \mathbb{N}$）
$d_{\widetilde{E}}(ee)$	$\widetilde{G}$の枝 eeの依存距離（$\widetilde{E} \rightarrow 0 \mathinner{..} N-1$）
$F_V(v)$	Gの節点 vの$\widetilde{G}$における投影元の節点の集合を返す関数（$V \rightarrow \mathbb{P}\, \widetilde{V}$）
$F_E(e)$	Gの枝 eの$\widetilde{G}$における投影元の枝の集合を返す関数（$V \rightarrow \mathbb{P}\, \widetilde{E}$）