NAIST-IS-DD0761018

博士論文

部分グラフの包含関係に着目した
代謝経路予測手法の提案

田中 健一

2010年 3月 17日

奈良先端科学技術大学院大学
情報科学研究科 情報生命科学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士(工学) 授与の要件として提出した博士論文である。

田中健一

審査委員：

金谷 重彦	教授	（主指導教員）
小笠原 直毅	教授	（副指導教員）
MD.ALTAF-UL-AMIN	准教授	（副指導教員）
高橋 弘喜	助教	（副指導教員）

部分グラフの包含関係に着目した代謝経路予測手法の提案*

田中 健一

内容梗概

人類は日常生活を送る上で食料、医薬品、香料、燃料等、様々な場面において多くの代謝物を利用している。構造決定されている代謝物は約50,000種と報告されており、KEGGでは16,021代謝物、KNApSAcKでは40,957代謝物がデータベース化されている。一方で、代謝経路に関しては、構造決定されている代謝物の約10%程度に相当する情報がデータベース化されるに留まっている。代謝経路の解明は生体による有用代謝物の高収率合成の実現にも繋がり非常に重要である。フィンガープリントや最大共通部分グラフを用いて化学構造間の類似性を評価し、代謝経路を予測する手法が提案されているが、予測精度の低さやNP困難なアプローチである等の課題が残っており、高速かつ予測精度の高いアルゴリズム開発が必須な状況である。本研究では、化学構造から最大頻出連結部分グラフを抽出し、部分グラフの包含関係に着目した代謝経路予測を行うことで予測精度を向上させた。また、既知代謝経路の大部分において基質と生成物間の骨格構造に包含関係が認められる点を確認し、この特徴をヒューリスティックとして導入することで、最大頻出連結部分グラフ抽出処理の高速化を実現した。先行研究では、15,050代謝物間の経路予測において計算を完了できたのが3.34%であったのに対し、提案手法では34,653代謝物間の経路予測が2週間程度で可能となった。また、予測結果から多くの既知代謝経路や新規代謝経路を確認することができた。さらに、代謝経路の視覚化を目的とし、自己組織化写像マップを応用した描画アルゴリズムを提案した。提案手法により、与えられた描画空間を有効に利用したネットワーク配置が可能となり、多くの化学構造を一枚の代謝マップ上に表示させることに成功した。開発したシステムはMetClassifierとしてWEB上で公開している。また、一部のアルゴリズムは公共データベースであるMassBankやNPEdiaの部分構造検索エンジンにも広く採用され現在稼働中である。

キーワード

代謝物，代謝経路予測，創薬，グラフ理論，ケモインフォマティクス

*奈良先端科学技術大学院大学 情報科学研究科 情報生命科学専攻 博士論文, **NAIST-IS-DD0761018**, **2010** 年 **3** 月 **17** 日.

Metabolic pathway prediction focusing on inclusive relation of subgraphs*

Kenichi Tanaka

Abstract

Metabolites are integral part of human life because of their usage as food, drug, spices, fuels etc. Chemical structures of about 50,000 metabolites have been determined and large amount of information about these metabolites are stored in many databases. KEGG organizes 16,021 metabolites with information of their metabolic pathways. KNApSAcK organizes 40,957 metabolites with species-metabolites relations. Though the chemical structures of around 50,000 metabolites are known information on their pathways is very limited. To understand and to be able to handle metabolic pathways is important to realize high-yield production of useful metabolites. Several approaches have been proposed for the prediction of metabolic pathways. But these approaches have some limitations e.g. low accuracy and/or high computational cost. In the present study, to improve the accuracy of metabolic pathway prediction, I focus on maximum frequent connected subgraphs among an input set of chemical structures and predict metabolic pathways by determining inclusive relation of such subgraphs. And for fast computation of the process of frequent subgraph extraction, I introduce a heuristic approach focusing on inclusive relation between frameworks of substrate and product. By using proposed approach, I predicted metabolic pathways among 35,000 metabolites in much shorter time compared to other approaches. I observe many known and new metabolic pathways in predicted result. To visualize predicted pathways, I propose a visualization algorithm called Network Self-Organization based on the concept of Self-Organizing map. Network Self-Organization can spread the entire pathway evenly to a given space. Developed algorithms have been implemented as integrated software named MetClassifier which is freely available on internet. Parts of the developed algorithms are used in public databases MassBank and NPEdia.

Keywords:

Metabolite, metabolic pathway, drug discovery, graph theory, chemo informatics

.* Doctoral Dissertation, Department of Bioinformatics and Genomics, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0761018, March 17, 2010.

目次

図目次

表目次

第I部　　代謝経路予測

第1章　序論

生物が生体内で合成する代謝物は日常生活のあらゆる場面で利用されており、非常に重要な役割を担っている。世の中には膨大な数の代謝物が存在し、その多くがデータベース化されているが、代謝物の合成経路である代謝経路に関しては大部分が未解明な状態にある。代謝経路を解明する為に数多くの手法が提案されてきたが、大量の代謝物を短時間で整理する決定的な手法は未だ提案されていない。本章では、初めに第 1.1 節において「代謝物が日常生活のどのような場面で利用されているのか」、「代謝経路がどの程度解明されているのか」という観点から代謝経路解明の必要性を述べる。次に第 1.2 節において代謝経路予測に関する先行研究を紹介し、代謝経路を全て解明することのできる決定的な手法が提案されていないことについて説明を行う。最後に第 1.3 節において本研究の目的を明確にする。

第1.1節　　研究背景

本節では、代謝経路解明の重要性を示す。第 1.1.1 項において社会および研究においてどのように代謝物が利用されているのかを述べ、第 1.1.2 項において代謝経路の大部分が未解明な状態にあることについて述べる。

第1.1.1項　　代謝物の重要性

人類は日常生活を送る上で食料、医薬品、香料、燃料等、様々な場面において多くの化合物を利用している。これらの化合物は主に天然からの取得、有機化学の手法のみを用いて合成する「全合成」、天然から取得した化合物を原料として有機化学の手法を適用した「半合成」等の手段を経て取得される。化合物の中でも生物から抽出される化合物は特に「代謝物」と呼ばれる。生物の生体内では、外界から取り込んだ物質を異化によって分解しエネルギーを得る反応と、同化によって生体に必要な代謝物を合成する反応が存在し、これら異化と同化をまとめて代謝反応と呼ぶ。代謝反応の中でも、多くの生物が共通に保有し生存するために必須な中心代謝系は一次代謝[1]と呼ばれる。一方で生物種が固有に持つ代謝は二次代謝[2]と呼ばれる。進化の過程で遺伝子に突然変異が起こることで偶然得た代謝が他の生物種との生存競争において有利に働いた場合、その生物種は自然淘汰されることなく生き残る。植物は動物のように動き回ることができない為、多様な二次代謝系を発達させることで生存競争に打ち勝ってきた。例えば、受粉に関わる昆虫が好む匂いや色素に関連

[1] 一次代謝の例：呼吸、光合成、アミノ酸の合成等
[2] 二次代謝の例：アルカロイド、フラボノイド、テルペノイドの合成等

する代謝物や、天敵に有害な代謝物を新規に得た場合に、他の生物種よりも生存する確率が高くなる。

構造が決定されている代謝物の総数は約 50,000 であると報告されている(Verpoote, 1998; De Luca and St Pierre, 2000)。また、構造が未決定な物も含むと約 200,000 の代謝物が存在するとの報告もある(Hostettmann, 2000)。これらの代謝物が医・薬・農・工学の様々な場面で利用されている。例えば、創薬の現場におけるリード化合物[3]の候補として、代謝物は重要な役割を果たす（Simmond and Grayer, 1999)。化学構造と薬理活性の間には相関があることが知られており、リード化合物の候補を見つける際、化学構造に多様性が認められる多種類の化合物が要求される(Valler and Green, 2000)。2003 年 Feher らは人工化合物、代謝物、薬に使われている化合物の化学構造の多様性比較結果を報告している、比較対象とした人工化合物の数(13,506)は代謝物の数(3,287)を上回っているが、化学構造の多様性に関しては代謝物の方が人工化合物より優れていることが分かる(Feher and Schmidt, 2003)(図 1)。

人工合成を用いた方法では、コンビナトリアルケミストリー[4]による大量合成が行われているが、代謝物と比較すると多様性のある化合物が合成できているとは言えない。また、目的化合物の合成に多段階の反応を経由した場合、収率の低下、反応に必要なエネルギーや使用する溶媒の増加等の問題が発生する。代謝反応の優れている点の一つとして、一度の代謝反応で化学構造が複雑な化合物の合成を実現していることも挙げられる。

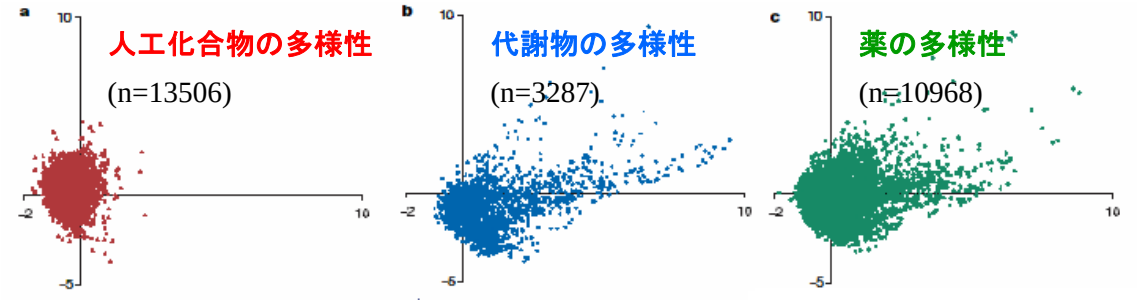


図 1 化合物の多様性調査結果

立体化学特性（光学異性体の中心、回転可能な結合、環状構造の結合度合）、電気特性（芳香族に関わる原子、電子を受け取る原子、電子を供給する原子）、構造特性（C-N 結合、C-O 結合、C-S 結合、C-ハロゲン結合）の計 10 個のパラメータを元に主成分分析を行った後、第一主成分と第二主成分を軸にして各化合物がグラフ上にプロットされている。

[3] 効力、選択性、薬物動態性、物理化学特性、毒性がない、新規性があると思われる化合物
[4] 複数の化学反応の組合せを変えることで膨大な種類の化合物を合成する手法、一連の作業は自動化、小型化、並列化され、短時間で膨大な種類の化合物を生成することが出来る。

第1.1.2項　　未解明な代謝経路

代謝物に関する情報をデータベース化する試みとして KEGG(Kanehisa and Goto, 2000)、KNApSAcK(Shinbo et al., 2006)等が挙げられる。2009 年 11 月の段階において KEGG には 16,021(13,841)代謝物、8,023 代謝経路に関する情報が登録されている。KNApSAcK は生物種と代謝物の関係をデータベース化したもので、40,957(34,653)代謝物、19,293 生物種、81,746(74,054)代謝物-生物種関連情報が登録されている。ここで、( )内の数字は化学構造情報も存在するデータ数である。構造が決定されている代謝物の総数は約 50,000 であると報告されており(Verpoote, 1998; De Luca and St Pierre, 2000)、代謝物に関するデータベースは充実していると言える。一方で、代謝経路に関する情報に関しては、KEGG に登録されている 8,023 代謝経路に出現する代謝物の数は 6,255 代謝物である。つまり、構造決定されている代謝物の約 90%に関しては、代謝経路が未解明な状態にあると言える。

代謝経路を解明することは、各生物種が進化の過程においてどのような代謝を獲得し、獲得した代謝が他の生物種との生存競争に打ち勝つ上でどのような役割を持っていたのかを理解することにも繋がり学術的価値は非常に高い。また、日常生活において利用価値の高い代謝物（もしくは化学構造が類似している代謝物）に関する代謝経路を解明し、さらに代謝反応にかかわる遺伝子を同定することができれば、遺伝子組み換え技術等を用いることで、目的代謝物の高収率合成の実現が期待できる。

第1.2節　　関連研究

本節では代謝経路の解明を試みた先行研究について紹介する。第 1.2.1 項では、文献情報や専門家の知見といった知識ベースを用いることで代謝経路の解明を試みた研究について述べる。第 1.2.2 項では、化学構造の類似性評価を行うことで代謝経路予測を試みた研究について述べる。

第1.2.1項　　知識ベースを用いた代謝経路の解明

知識ベースを用いた代謝経路の解明を試みた先行研究として、大量の文献情報を基にした代謝反応の整理(Kaichi, 2008)、化学構造の類似性に着目した代謝経路予測(Tokimatsu and Arita, 2006)、代謝反応ルールベース構築(Langowski and Long , 2002; Talafous et al., 1994; Ellis et al., 2006; Hou et al., 2004; Oh et al., 2007)等が過去に提案されている。

本研究における共同研究者である海内は alkaloid 代謝経路の体系化を目的とし、アカデミックプレス社より発行された「The Alkaloids」1～63 巻(1949 年～2006 年)、および天然物化学における総説「Natural Product Report」(1984 年～2007 年)を調査対象とし、453 の alkaloid 代謝物からなる 540 対の代謝二項関係を抽出し、体系化を行った(Kaichi, 2008)。文献から得られる代謝経路情報の多くは、研究対象に関連する代謝反応である為、代謝反応全体からみた場合断片的な情報となる。これら過去に報告された膨大な代謝反応に関す

る文献情報を整理し、代謝反応を繋ぎ合せることで、代謝反応全体像の理解を試みた(図 2)。

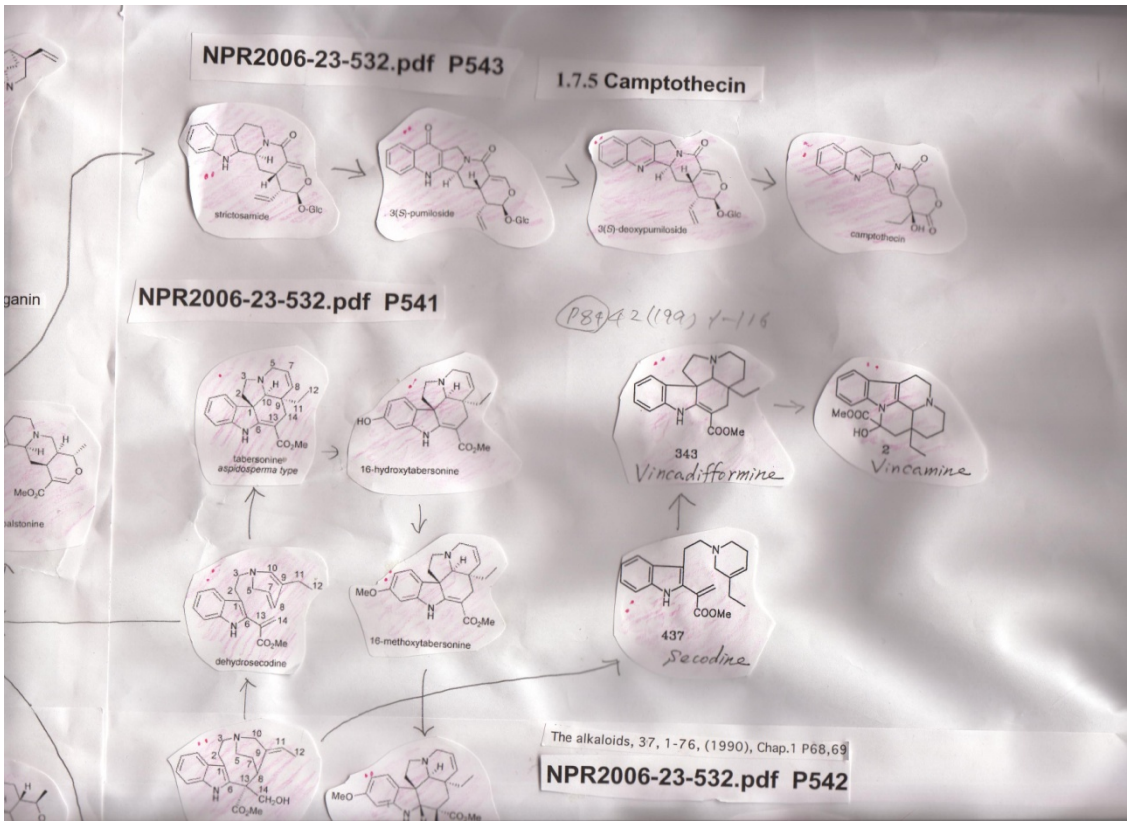


図 2 文献情報を基にした alkaloid 代謝反応に関する体系化結果の一部

複数の文献から得られた代謝反応情報を一枚の巨大マップに整理している。事前に代謝反応の全体像は把握できない為、各化合物を紙に印刷し、試行錯誤を重ねて代謝物の位置を決定することで体系化が行われている。代謝マップ全体を PC や論文の紙面上に表現しようとした場合、面積の制約から代謝物を ID で表現することになる為、代謝経路全体に渡り化学構造がどのように変化していくのかを認識することが難しい。巨大な紙面上に化学構造を直接配置することで、構造情報に基づいた代謝経路の全体像が把握できた。

生化学の専門家の視点から代謝物を整理する試みとして、時松らによる Flavonoid Viewer が挙げられる(Tokimatsu and Arita, 2006)。Flavonoid Viewer では KNApSAcK に収録されている代謝物の内、フラボノイドに分類される代謝物に関し、化学構造の特徴に基づいた階層分類を行うことで生合成経路と代謝産物の対応付けが行われている。Flavonoid Viewer では、12 桁からなる ID(図 3)を用いて分子構造を表現し、フラボノイドを階層分類している(図 4)。

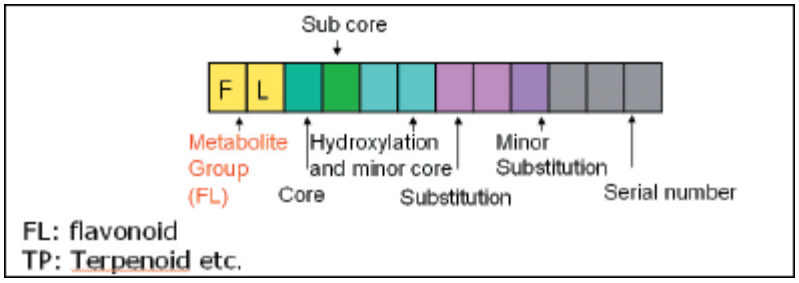


図 3 Flavonoid Viewer で用いられている 12 桁の ID

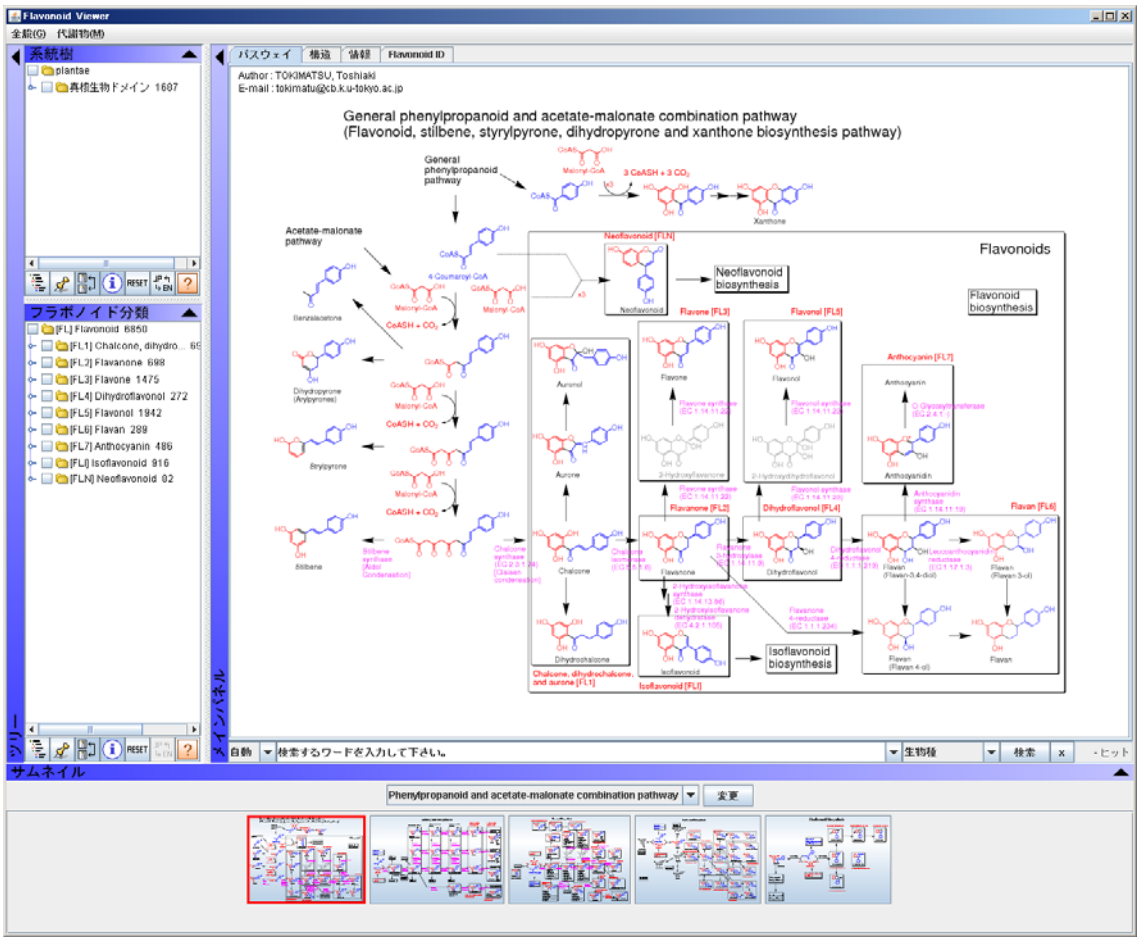


図 4 Flavonoid Viewer

既知の代謝経路から頻出する反応ルールを抽出しデータベース化することで、未知の代謝反応を予測した研究例がある(Langowski and Long , 2002; Talafous et al., 1994; Ellis et al., 2006; Hou et al., 2004; Oh et al., 2007)。例えば、ミネソタ大学で運営されているUM-BBD では、主に人工合成物質の微生物異化に用いられてきた化合物、酵素、反応、パスウェイに関するデータを収集公開している。また、収集した代謝反応から共通反応ルールを抽出し、専門家の視点から各反応ルールの起こりやすさをランク付けすることで反応ルールベースを構築し、これを用いることで代謝反応予測を行う Pathway Prediction System (PPS)も公開している(Ellis et al., 2006)。この中で例えば、図 5 に示した反応ルール(bt0353)では、「オルト位とパラ位においてヒドロキシル化される反応は起こりやすい」という反応ルールが定義されている。図 6 は Pathway Prediction System による代謝経路予測の例である。WEB 画面を通して入力された化学構造に対して、bt0001, bt0353, bt0270 の 3 つの反応ルールを適用することで得られる可能性のある化学構造を予測している(図 6 (左))。さらに予測した化学構造を選択することで、選択した化学構造に対し反応ルールが適用される(図 6 (右))。

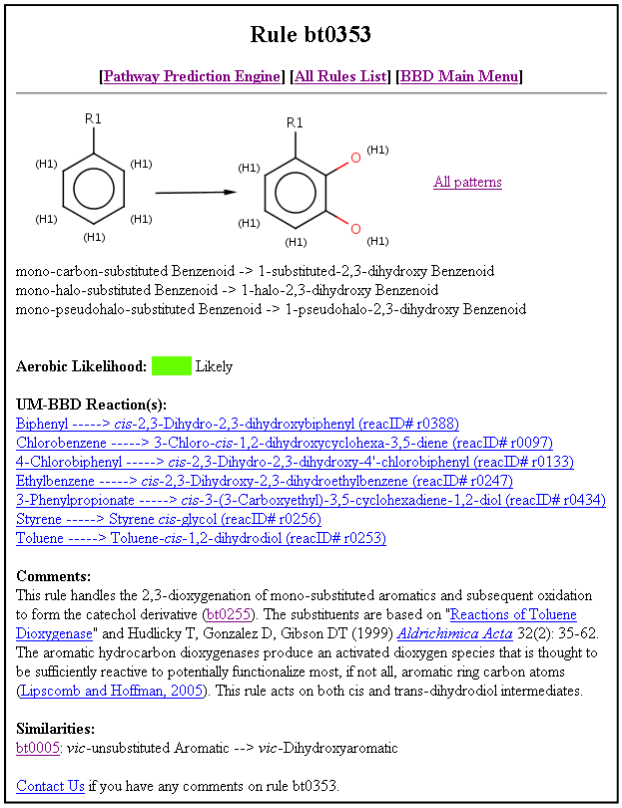


図 5 反応ルールの例(bt0353)
この反応ルールでは、「オルト位とパラ位がヒドロキシル化される反応は起こりやすい」というルールが定義されている。

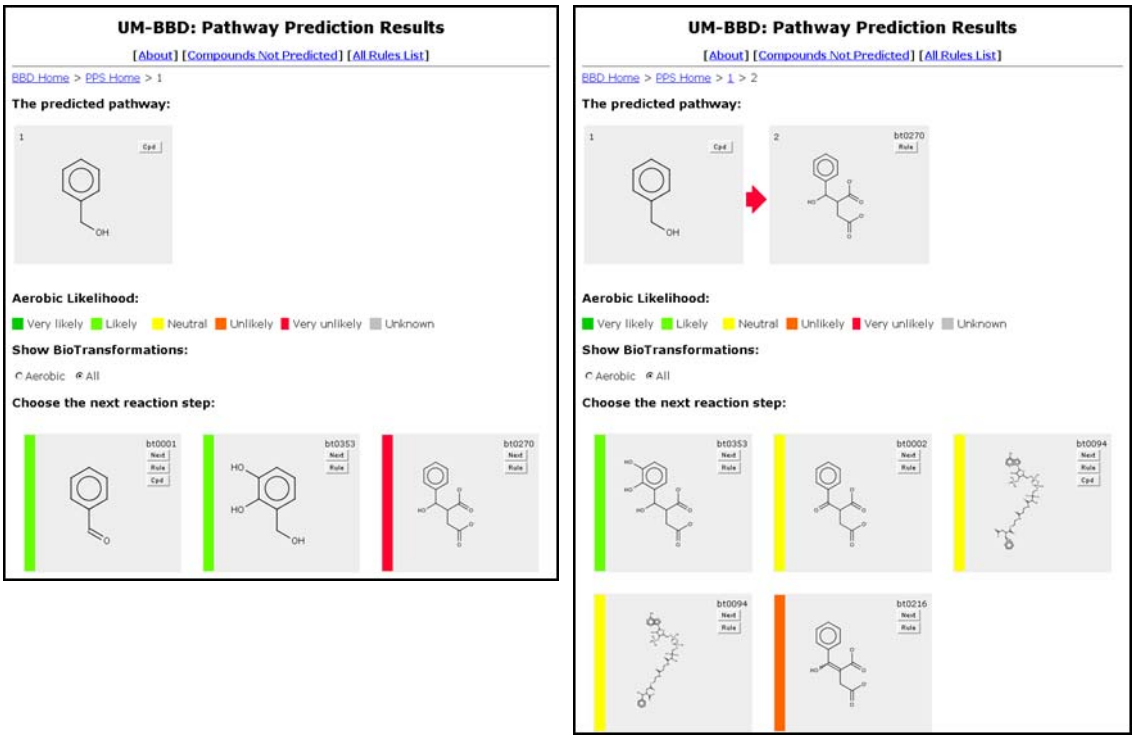


図 6 UM-BBD: Pathway Prediction System による代謝経路予測
WEB 画面を通して入力された化学構造に対し、適用可能な反応ルールを検索し、反応ルール適用後の化学構造を出力する。図 6(左)は入力された化学構造に対して、bt0001, bt0353, bt0270 の 3 つの反応ルールを適用することで得られる化学構造を予測している。予測した化学構造を選択すると、選択した化学構造に対し反応ルールが適用される(図 6(右))。

第1.2.2項　　化学構造の類似性評価による代謝経路予測

第 1.2.1 項で述べた知識ベースを用いた代謝経路予測結果から得られる情報は、「専門化が何に着目しているのか」、「どのような情報が重要であるのか」等を判断する指標となる。しかし、知識ベースを用いた代謝経路予測の能力は知識ベース構築時に用いたデータセットのサイズと種類に依存する。また、自然界に存在する代謝物の数は膨大であり、これら全ての反応をマニュアルオペレーションで分類することは非常に困難な作業である。

この問題を解決する為、知識ベースを用いる代わりに化学構造の類似性を評価することで代謝経路予測を行う手法が提案されている。化学構造の類似性を評価する手法の多くは、定量的構造活性相関の分野において数多く提案されており、これらの手法が代謝経路予測に応用されている。類似性評価手法はおおまかに Fingerprint-based approach と Graph-based approach に分けることが出来る。Fingerprint-based approach による類似性

評価では、事前に Fragment library[5]と呼ばれる部分構造のセットを定義し、各化合物がどの部分構造を保有しているかを Fingerprint(Lewis et al., 2000; Xue et al., 2003)と呼ばれる 0,1 からなるビット列で表現する(図 7)。その後、2 つの Fingerprint 間の類似係数を計算する。類似係数として、最もよく用いられるのは Tanimoto 係数である(Willett et al., 1998)。Tanimoto 係数は全体に対する共通部分の割合を数値化した係数であり、以下の計算式で値を求めることが出来る(図 8)。

$$\text{Tanimoto 係数} = \frac{\text{共通部分の大きさ}}{\text{共通部分の大きさ} + \text{相違部分の大きさ}}$$

Fingerprint-based approach を用いた代謝経路解析として、複数の既知代謝経路について代謝経路上の化学構造の類似性を評価することで、類似代謝経路を抽出する研究などが行われている(Tohsato and Nishimura, 2008)。しかし、Fingerprint-based approach は、2 つの化学構造間の類似性を高速に計算することが可能である反面、Fragment 間の連結性が無視されてしまう為、曖昧な評価になる点が指摘されている(Raymond et al., 2002)。

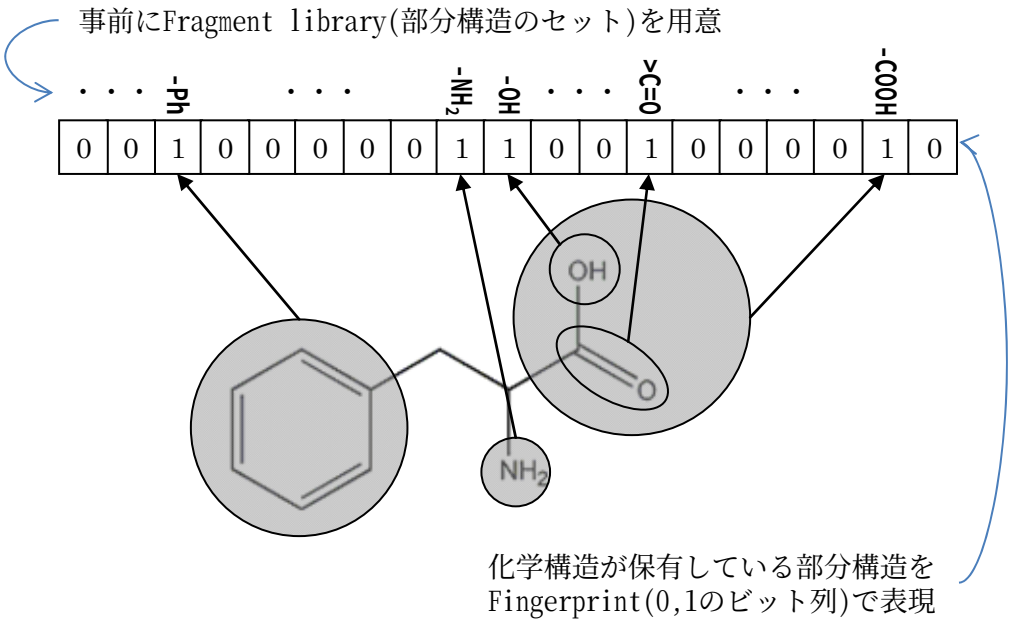
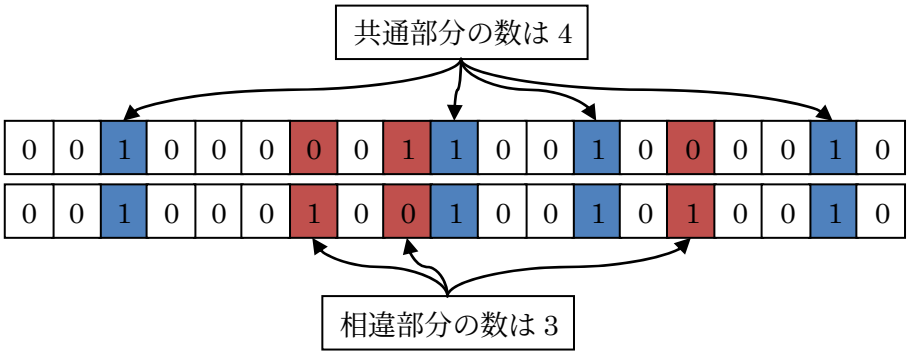


図 7 Fingerprint による化学構造の表現

[5] Fragment library としてよく用いられているのは Molecular Design Limited 社（現在の Symyx 社の前身）が提供している MACCS key (http://www.symyx.com/downloads/)が挙げられる。MACCS key からは 166bits の Fingerprint を得ることが出来る。



$$\text{Tanimoto 係数} = \frac{4}{4+3} = \frac{4}{7}$$

図 8 Fingerprint から得られる類似係数(Tanimoto 係数)
共通部分（両方の Fingerprint が 1 となっている部分）は 4 箇所、相違部分（片方の Fingerprint が 1 となっている部分）は 3 箇所。この時、Tanimoto 係数は 4/(4+3)=0.57 と計算できる。

Graph-based approach による類似性評価は、類似係数を用いる点は Fingerprint-based approach と同じである。異なる点は、共通部分や相違部分の大きさを求める際に、2 つの化学構造が共通に保有する最大共通部分グラフ（Maximum Common subgraph)を取得した後、Maximum Common subgraph に含まれる要素(原子や結合)を共通部分グラフ、含まれない要素を相違部分グラフとして扱う点である。Fingerprint-based approach では、Fingerprint を得る際、化学構造が Fragment library に含まれるどの部分構造を含んでいるのかを調べればいいので、化学構造の複雑さに影響を受けることなく Fragment library に含まれる部分構造の数だけ部分構造検索を実行すればいい。一方で、Maximum Common subgraph を取得する問題は、根本的には片方の化学構造から取得可能な全ての部分構造に対し、もう片方の化学構造に含まれる共通部分構造であるかを調べ、さらに共通部分構造の中から最もサイズの大きな部分構造を取得する問題になる。ここで、一つの化学構造から取得可能な部分構造は化学構造に含まれる結合の組合せの数($2^{\text{結合の数}}$)だけ存在する。この為、全ての部分構造を対象として Maximum Common subgraph を取得する問題は、NP 困難[6]なアルゴリズムとなり、複雑な化学構造を対象とした場合、膨大な計算コストが要求される。Maximum Common subgraph を取得するアルゴリズムとして有名なのは 2002 年

[6] NP 困難(NP(Non-deterministic Polynomial)-hard problem)： 計算対象の数が 10 個の時に 100 分、1,000 個の時に 1,000,000 分、n 個の時に n^2 分だけの計算時間が必要な計算処理の複雑さは $O(n^2)$と表現し、n^2 分や n^3 分を多項式(Polynomial)時間と呼ぶ。NP-hard problem というのは多項式時間で計算が完了しない複雑な問題（例えば、2^n 分かかるような問題）を意味する。今回のケースで、n 個の結合を持つ化学構造からは 2^n 個の部分構造が取得可能であり、この中から Maximum Common Subgraph を探索する問題に相当する為、NP 困難な問題に相当する。

に Raymond らが提案した RASCAL である(Raymond et al., 2002)。RASCAL では、Maximum Common subgraph を探索する前処理として、比較対象の 2 つの化合物の構造式や化学構造に含まれる各原子の結合次数、各結合の両端の原子等に着目し、Maximum Common subgraph の探索を行う際に完全解が得られる保証を残す打ち切り限界を事前に設定し、Maximum Common subgraph になりえない部分構造を探索対象から除外することで検索速度の向上を実現した。しかし、RASCAL のようなアルゴリズムを用いても、化学構造が複雑な化合物に関しては、膨大な計算時間が必要になり、ヒューリスティック[7]の導入は必要であると指摘されている(Hattori et al., 2003)。2008 年小寺らは、解析対象とする化学構造間の Maximum Common subgraph を取得した後、化学構造間の Tanimoto 係数を類似係数として計算することで代謝経路を予測する手法を提案した(Kotera et al., 2008)。彼らの提案手法では、KEGG に登録されている 15,050 代謝物の全組合せ 113,243,725(=15,050 × 15,049 ÷ 2)から、H2O や CO2 の様に構造が単純なケース(35,636,614 ペア)、および C3H7NS と C6H7O4P の様に非水素原子の数が 5 以下のケース(2,840,140 ペア)、計 38,476,754 ペアを除外した 74,766,971 ペアを比較対象としている。しかし、論文の中で計算が完了したとされているのは比較対象とした 74,766,971 ペアの 3.34%にあたる 2,502,333 ペアである。比較対象の化学構造ペア両方が複雑な環状構造を含むケースにおいて計算が完了しなかった。

第1.3節　　研究目的

以上に述べたように、代謝物は現代社会において非常に重要な存在である。論文として報告されている代謝物の化学構造情報に関しては KEGG や KNApSAcK のようなプロジェクトによってデータ整備が進められている。しかし、代謝経路が判明しているのは、その内の 10%程度についてのみである。これらの代謝物に関する代謝経路を予測する為の手法がいくつか提案されてきたが、数万代謝物に関する代謝経路を高速かつ正確に予測する手法は未だ提案されておらず、この問題を解決することが本研究の目的である。

[7] 厳密解を求めようとはせず、経験則等を根拠に検索対象の削減や探索の打ち切りを行うことで、計算速度の向上を目的するアプローチ

第2章　Graph theory

本論文における提案手法は Graph-based approach を用いて代謝経路予測を行う。そこで、本章では提案手法の中で用いている Graph theory についてまとめる。まず、第 2.1 節において用語の定義を行い、第 2.2 節において定義から導くことのできる定理について述べる。ここで述べる定理は提案手法の中で計算コストの低減に利用している（第 4 章)。第 2.3 節では、提案手法で着目している Connected subgraph の効率的な探索方法の説明を目的として、Frequent Connected subgraph の抽出に関する先行研究の中から、提案手法にアプローチが近い gSpan(Yan and Han, 2002)の紹介を行う

第2.1節　　定義

[定義 1] Chemical graph

化学構造に含まれる原子を node、結合を edge としてグラフ表現したものを Chemical graph と定義する。

[定義 2] Subgraph

Chemical graph に含まれる node と edge の部分集合から得られるグラフを Subgraph と定義する。

[定義 3] 包含関係の表現

Subgraph（Chemical graph）A に含まれる node と edge の部分集合から Subgraph（Chemical graph）B が構成できる場合、「B ⊂ A」と表現する。

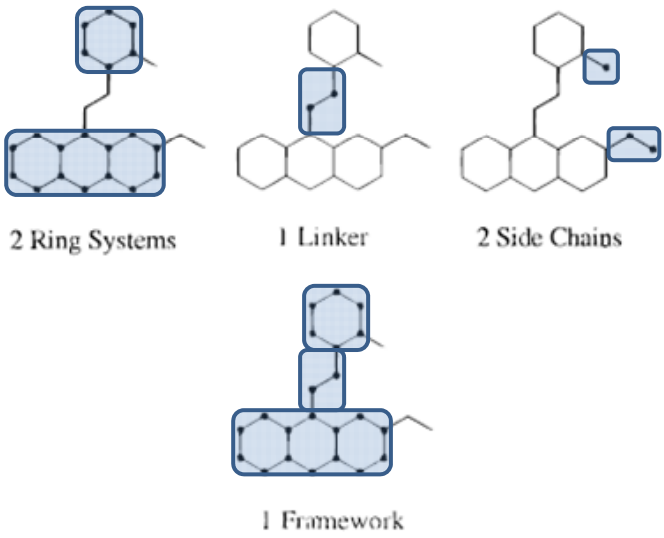


図 9 Bemis らによる化学構造記述子（Bemis and Murcko, 1996）

[定義 4]～[定義 6]では単一の Chemical graph から取得可能な Subgraph に対し、構造の特徴から定義される Subgraph について説明する。

[定義 4] Connected subgraph

全ての node と edge が連結している Subgraph を Connected subgraph と定義する。

[定義 5] Ring, Linker, Side chain, Framework(Bemis and Murcko, 1996)

- Ring System

  環構造および環構造に含まれる辺を共有して連結している環構造群を 1 つの Ring System として定義する。図 9 の場合、6 つの原子から構成される Ring System と 3 つの環から構成される Ring System の計 2 つの Ring System が存在する。

- Linker

  Ring System 間の連結にかかわっている原子を Linker Atom と定義する。図 9 の場合、2 つの Linker Atom から構成される Linker が 2 つの Ring System を連結している。

- Side Chain

  Ring System, Linker Atom のどちらにも所属しない原子を Side Chain Atom と定義する。図 9 の場合、1 つの Side Chain Atom から構成される Side Chain が 6 つの原子から構成される Ring System に結合しており、2 つの Side Chain Atom から構成される Side Chain が 3 つの環から構成される Ring System に結合している。

- Framework

  Ring System と Linker の和を取った構造を Framework と定義する。図 9 の場合、2 つの Ring System と 1 つの Linker から Framework が構成されている。

[定義 6] Framework-based Connected subgraph

Connected subgraph が、Subgraph の取得元である Chemical graph の Framework を含んでいる場合、Framework-based Connected subgraph であると定義する。元の Chemical graph が Framework を持たない鎖状構造の場合、全ての Connected subgraph は Framework-based Connected subgraph であると定義する。

[定義 7]～[定義 11]では複数の Chemical graph から取得可能な Subgraph に対し、Subgraph 間の包含関係やいくつの Chemical graph に含まれるか(出現頻度)等の条件から定義される Subgraph について説明を行う。

[定義 7] Common subgraph

2 つの Chemical graph に共通に含まれる Subgraph を Common subgraph と定義する。

[定義 8] Maximum Common subgraph

Tanimoto 係数等を用いて 2 つの Chemical graph の類似度を算出する事を目的として

Maximum Common subgraph を用いる場合には、Common subgraph の中でも最も要素数(node や edge の数)の多い Subgraph が Maximum Common subgraph と定義されることが多い。本研究では「Common subgraph A ⊂ Common subgraph B となる Common subgraph B が存在しない場合、Common subgraph A は Maximum Common Subgraph である」と定義する。

[定義 9] Frequent subgraph

複数の Chemical graph に多く含まれる Subgraph を Frequent subgraph と定義する。「Frequent subgraph は Multiple Common subgraph である」と考えることもできるが、Common subgraph と異なる点は、抽出対象の全ての Chemical graph に含まれる必要がない点である。複数の Chemical graph から Frequent subgraph を抽出する際には、事前に最低出現頻度（いくつの Chemical graph に含まれるか）を設定する。例えば、10 個の Chemical graph から Frequent subgraph を抽出する際、最低出現頻度を 3 と設定した場合、3 個以上の Chemical graph に含まれる Subgraph が Frequent subgraph となる。

[定義 10] Maximum Frequent Connected subgraph

出現頻度が等しい Connected subgraph の中で Subgraph A ⊂ Subgraph B となる Subgraph B が存在しない場合、Subgraph A を Maximum Frequent Connected subgraph と定義する。本論文では MFCS と省略した表記を用いる。

[定義 11] Framework-based Maximum Frequent Connected subgraph

Framework-based Connected subgraph のみを対象とした MFCS を Framework-based Maximum Frequent Connected subgraph と定義する。本論文では Framework-based MFCS と省略した表記を用いる。

[定義 12]および[定義 13]は、複数の Subgraph に対し Subgraph 間の包含関係の比較によって得られる、相対的な Subgraph の呼び方について定義を行う。

[定義 12] Ancestor subgraph, Descendant subgraph

Subgraph A⊂Subgraph B を満たす場合、

- Subgraph A は Subgraph B の Ancestor subgraph であると定義する。
- Subgraph B は Subgraph A の Descendant subgraph であると定義する。

[定義 13] Parent subgraph, Child subgraph

[定義 12]を満たし Subgraph A⊂Subgraph C⊂Subgraph B となる Subgraph C が存在しない場合

- Subgraph A は Subgraph B の Parent subgraph であると定義する。
- Subgraph B は Subgraph A の Child subgraph であると定義する。

※Parent subgraph は Ancestor subgraph の部分集合になる。

※Child subgraph は Descendant subgraph の部分集合になる。

第2.2節　　定理

[定理１] Subgraph A ⊂ Subgraph B の時、以下が成立する。

- Each Ancestor subgraphs of Subgraph A ⊂ Subgraph B
- Subgraph A ⊂ Each Descendant subgraphs of Subgraph B

Subgraph A の出現頻度が k の時、[定理２]～[定理５]が成立する。

[定理２] Subgraph A ⊂ Subgraph B ならば Subgraph B の出現頻度は k 以下になる。
[定理３] Subgraph B ⊂ Subgraph A ならば Subgraph B の出現頻度は k 以上になる。
[定理４] Subgraph A の Descendant subgraph の出現頻度は k 以下になる。
[定理５] Subgraph A の Ancestor subgraph の出現頻度は k 以上になる。

[定理６] 出現頻度が 1 の Subgraph の中で MFCS と成りうるのは、Subgraph の抽出元となっている Chemical graph の全ての node と edge から構成される Subgraph のみである。

[定理７] 2 つの Framewrok-based Connected subgraph A, B について、以下に述べる条件は、A ⊂ B が成立する為の必要条件である。

- Framework of A ⊂ Framework of B
- # of edges of A ≦ # of edges of B
- # of nodes of A ≦ # of nodes of B

ここで、# of edge of A は Framewrok-based Connected subgraph A に含まれる edge の数を表す。

第2.3節　　Frequent Connected subgraph 抽出に関する先行研究

提案手法は、代謝経路予測の過程において Framework-based MFCS の抽出を行う。Framework-based MFCS 抽出アルゴリズムは、「ユニークな Connected subgraph を探索しながら、他の Subgraph と出現頻度を比較することで目的の Subgraph を取得する」点において、データマイニングの分野において提案されている Frequent Connected subgraph を抽出するアルゴリズムに処理内容が近い(Huan et al., 2003; Kuramochi et al., 2001; Inokuchi et al., 2000; Inokuchi et al., 2002; Inokuchi et al., 2003; Yan and Han, 2002; Nijssen et al., 2004)。本節では、提案手法に特にアプローチが近い gSpan(Yan and Han, 2002)について紹介する。最近の研究では、RNA の頻出ステムパターンのマイニング(Hamada, 2008)や、頻出する糖鎖パターンの発見(Takigawa et al, 2009)等にも応用されている。

第2.3.1項　　gSpan－基本概念および定義－

複数の Graph から Frequent subgraph を取得する問題は、根本的には入力とした複数の Graph から取得可能な全ての Subgraph に対し出現頻度(いくつの Graph に含まれるか)を計算する処理になる為、Maximum Common subgraph を抽出する問題と同様に NP 困難なアルゴリズムになる。gSpan では対象とする Subgraph を Connected subgraph のみに限定することで、計算コストを低減している。図 10 において Graph(E, V), V = {v_0, v_1, v_2, v_3}, E = {v_0v_1, v_0v_2, v_1v_2, v_2v_3}から Connected subgraph を抽出するアルゴリズムを示す。ここで、node v_0, v_1, v_2, v_3 にはそれぞれ、A, B, B, C のラベルが付けられており、edge v_0v_1, v_0v_2, v_1v_2, v_2v_3 にはそれぞれ a, b, b, d のラベルが付けられている。Connected subgraph のみを対象とする為には、まず、入力とした Graph に含まれる edge のどれか 1つを初期 Subgraph とする(Subgraph←one edge of Graph)。Graph 上で初期 Subgraph に採用した edge に接続している別の edge を、初期 Subgraph に追加することで新しい Subgraph を得る。(New Subgraph←Subgraph∪Connected edge)。Graph 上で探索中の Subgraph に接続している edge を追加する作業を繰り返すことで Connected subgraph のみを得ることを保証する。しかし、この方法だけでは、edge を追加する順番を変えることで同一の Subgraph が複数のルートから発生することになり無駄な探索が発生する(図 10)。

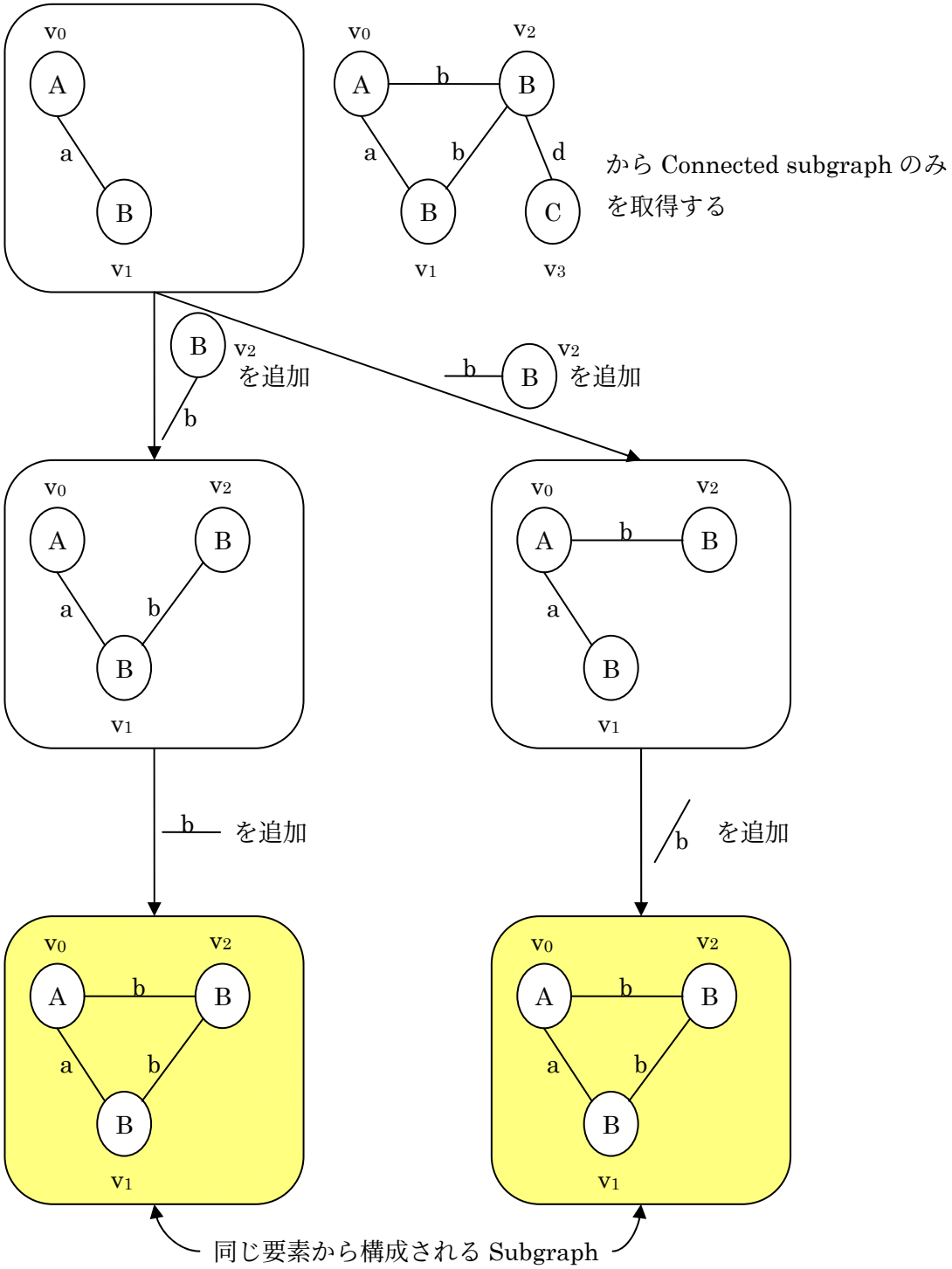


図 10 Connected subgraph の取得

1 つの edge からなる Subgraph を起点とし、Graph 上で Subgraph に隣接している edge を追加する操作を繰り返すことで、Connected subgraph のみを取得出来る。しかし、この方法では、隣接 edge を追加する順番を変えることで同一の Subgraph が複数のルートから発生することになり無駄な探索が発生する。

Unique Connected subgraph のみを探索する為に、gSpan では以下に説明する DFS Tree、DFS Code、DFS lexicographic order（DEF 辞書順）、Minimum DFS Code の概念を定義している。Frequent connected subgraph を取得する前処理として各 graph の Minimum DFS Code を取得する。隣接 edge を追加することで Connected subgraph を取得する度に、Minimum DFS Code との比較を行い、探索を打ち切るか継続するかどうかの判定を行うことで無駄な探索の回避を行っている。詳細なアルゴリズムに関しては、第 2.3.2 項にて説明を行う。

- DFS Tree

  Graph に含まれる全ての node と edge を DFS（Depth-First Search：深さ優先探索）を用いて探索した場合、複数の経路で探索することが可能である。図 10 における Graph をもとに DFS による探索を行った場合の例を図 11 に示す。DFS の過程で見つかった node 順で node のラベルを $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$ と付け直した後、ラベル順で node を階層表示することで、DFS Tree として表現することが出来る(図 12)。ここで、図 12 の赤い edge はすでに Subgraph に含まれている node 間を新たに接続した edge を表現している。

- DFS Code

  DFS による探索の過程で通過した edge を（始点 node の見つかった順、終点 node の見つかった順、始点 node のラベル、edge のラベル、終点 node のラベル）の組合せで表記した探索経路の表現を DFS Code と定義する。図 12 の左上の Graph について、4 個の node v_0, v_1, v_2, v_3 には、それぞれ A, B, B, C のラベル付けられており、edge v_0v_1, v_0v_2, v_1v_2, v_2v_3 にはそれぞれ a, b, b, d とラベル付けられている。ここで、得られる DFS Code の例として(0, 1, A, a, B)とは、node v_0, v_1 が A と B、edge v_0v_1 が a とラベル付けされていることを意味する。同様に(1, 2, B, b, B)とは node v_1, v_2 が共に B、edge v_1v_2 が b とラベル付けされていることを意味する。

- DFS lexicographic order

  （edge のラベル、始点 node のラベル、終点 node のラベル）の順に辞書順で各 DFS Code を順位付けした結果を DFS lexicographic order と呼ぶ。例えば、図 12 の 3 つのグラフに対する DFS Code において対応するラベルはそれぞれ、α ={(a, A, B), (b, B, B), (b, B, A), (d, B, C)}、β ={(a, B, A), (b, A, B), (b, B, B), (d, B, C)}、γ ={(d, B, C), (b, B, A), (a, A, B), (b, B, B)}である。まず、各 DFS Code の先頭要素{α :(a, A, B), β :(a, B, A), γ :(d, B, C)}の比較を行う。edge のラベルの比較により、α , β (a)→ γ (d)が確定する。順序が確定しなかったα , β に関しては始点 node のラベルの比較により、α (A)→ β (B)が確定し、DFS lexicographic order は$\alpha \rightarrow \beta \rightarrow \gamma$ となる。始点ノードのラベルの比較でも順位が確定しない DFS Code が存在した場合、終点ノードのラベル比較を行い。それでも順位が確定しない DFS Code が存在した場合、2 番目の要素、3 番目の要素と順に比較を行うことで DFS lexicographic order を決定する。

- Minimum DFS Code

  全ての DFS Tree から得られる DFS Code を DFS lexicographic order で並べたときに最小となる DFS Code を Minimum DFS Code と定義する。図 12 のケースではαが Minimum DFS Code になる。

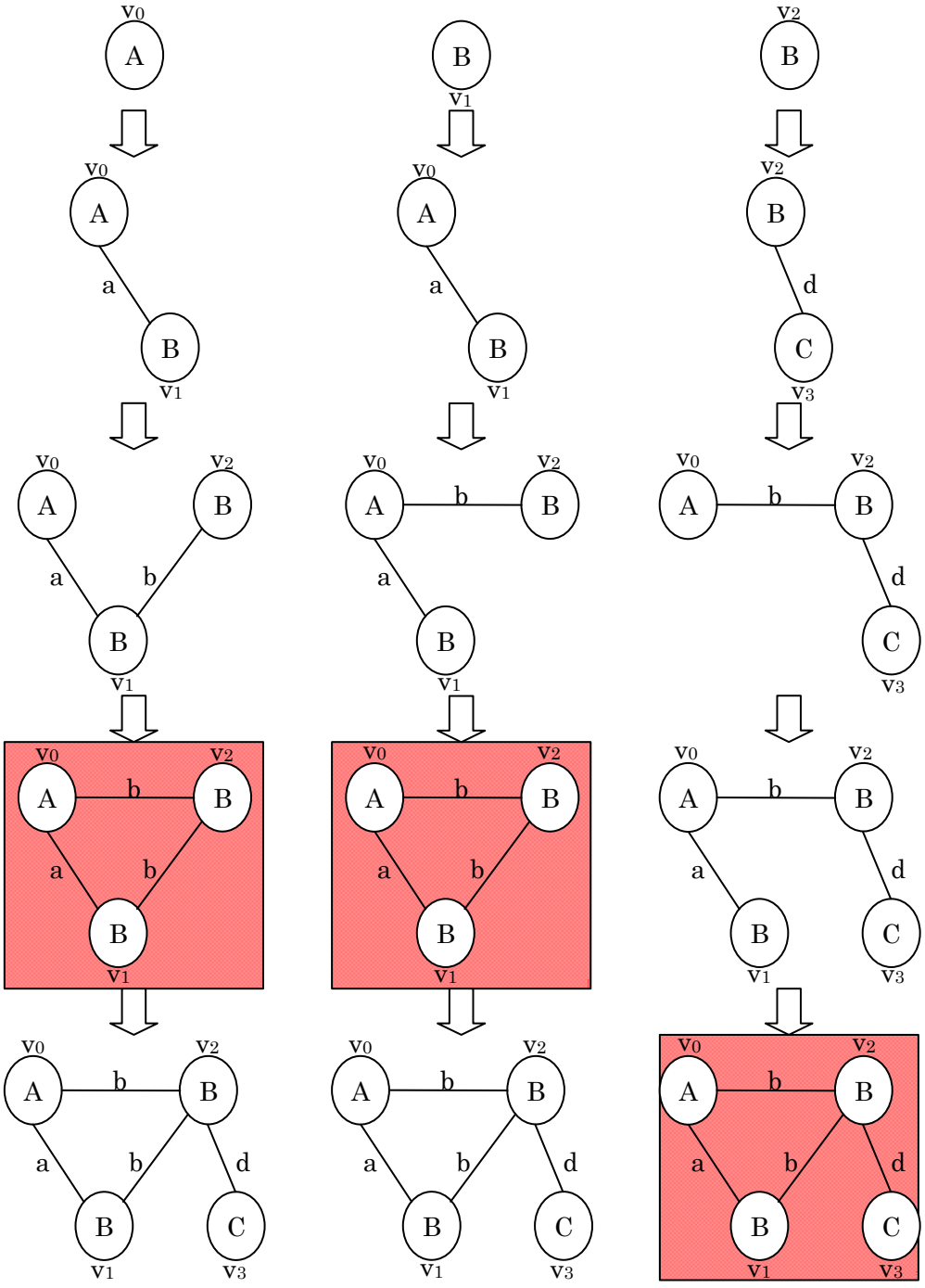


図 11 複数存在する深さ優先探索の探索順序

背景色が付いている Connected subgraph は edge のみの追加により得た Subgraph である。

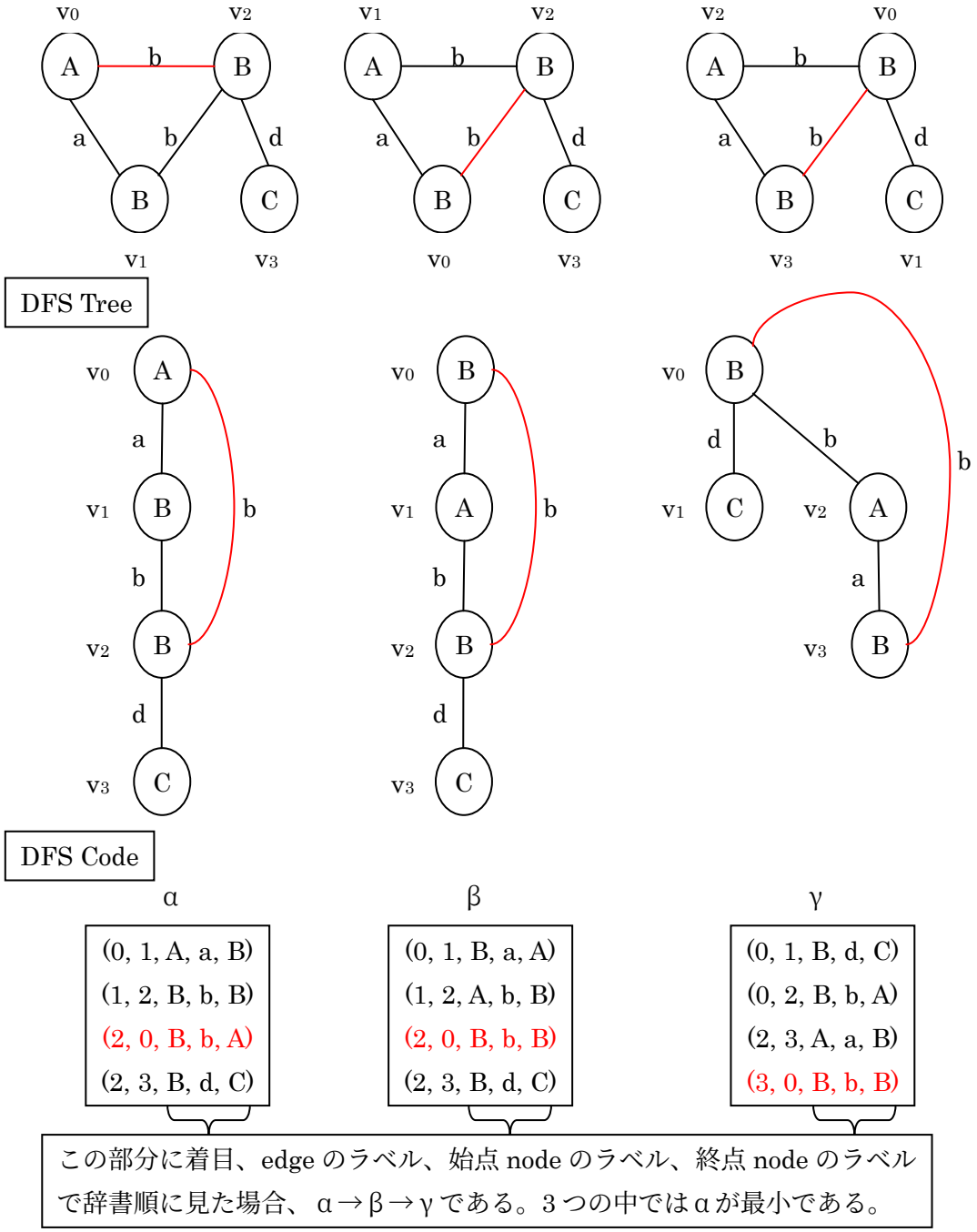


図 12 図 11 の 3 つの深さ優先探索に対応した、DFS Tree および DFS Code

DFS で見つかった node 順に v_0 → v_1 → v_2 → v_3 とラベルを付け直す。赤い edge はすでに Subgraph に含まれている node 間を新たに接続した edge を表現している。DFS Code は(始点 node の見つかった順、終点 node の見つかった順、始点 node のラベル、edge のラベル、終点 node のラベル)で構成されている。(edge のラベル、始点 node のラベル、終点 node のラベル）の順に辞書順で各 DFS Code に順位付けをすると α→β→γ となる。全ての DFS Code から最小 DFS Code を見つける。※今回のケースでは α が最小 DFS Code

第2.3.2項　　gSpanーアルゴリズムの説明ー

gSpan のアルゴリズムについて説明する。gSpan の疑似コードを図 13 に示す。
入力変数の定義は以下の通りである。

D：入力とした graph のセット

S：Frequent Connected subgraph

Algorithm 1 の 1～3 行目では、入力 graph に含まれる各要素(node と edge)に着目し、最低出現頻度(MinSup)以下の要素を graph から取り除く。例えば、図 14 のケースを例にすると、node C は 2 つの graph に含まれ、node D は 1 つの graph にしか含まれない。ここで、仮に最低出現頻度を 3 と設定した場合、node C や node D を含む全ての Subgraph は Frequent Connected subgraph の対象外である。事前にこれらの要素を削除することで graph が単純になり、抽出可能な Connected subgraph の数が減少する為、この後の処理における計算コストが低減する。4～6 行目は graph に含まれる edge を探索の初期 Subgraph S1 とする。また、1～3 の処理で排除されなかった edge を Frequent Connected subgraph S に追加する。7～10 行目では、S1 に含まれている edge e を初期 Subgraph s として用い(8 行目)、Frequent Connected subgraph の探索を開始する(9 行目)。9 行目の処理は初期 Subgraph S1 に採用した edge を含む全ての Connected subgraph から Frequent Connected subgraph の抽出が完了した段階で終了する。S1 に含まれる別の edge を初期 Subgraph s として用い、Frequent Connected subgraph の抽出を行う際には、過去に初期 Subgraph s に用いた edge を含む全ての Subgraph は探索済みの Connected subgraph になる為、10 行目で探索が完了した edge を graph から削除することで無駄な探索を回避する(10 行目) (図 15)。

Algorithm 1 における 9 行目の Subgraph_Mining(D, S, s)について Subprocedure 1 により説明する。1～2 行目が gSpan で重要な部分である。事前に作成した Minimum DFS Code と、探索経路から得られた DFS Code を比較し、DFS Code が Minimum DFS Code から生成できない DFS Code であった場合、探索を打ち切る(図 16)。探索を打ち切られることなく 3 行目に到達した Subgraph は Frequent Connected subgraph であることが確定するので S に追加する。4～8 行目では Subgraph s の Child subgraph c が最低出現頻度を満たすかどうかを判定し、満たさない場合は探索を打ち切る(6 行目)。満たしていた場合は Child subgraph c を用いて再帰的に Connected subgraph を探索する(図 17)。

以上の処理によって、gSpan では重複のないユニークな Connected subgraph のみを探索し、各 Connected subgraph の出現頻度を部分構造検索によって調べることで、Frequent Connected subgraph の抽出を行う。

Algorithm 1 GraphSet_Projection($\mathbb{D}$,$\mathbb{S}$).

1: sort the labels in $\mathbb{D}$ by their frequency;
2: remove infrequent vertices and edges;
3: relabel the remaining vertices and edges;
4: $\mathbb{S}^1 \leftarrow$ all frequent 1 edge graphs in $\mathbb{D}$;
5: sort $\mathbb{S}^1$ in DFS lexicographic order;
6: $\mathbb{S} \leftarrow \mathbb{S}^1$;
7: **for each** edge $e \in \mathbb{S}^1$ **do**
8: initialize s with e, set $s.D$ by graphs which contains e;
9: Subgraph_Mining($\mathbb{D}$, $\mathbb{S}$, s);
10: $\mathbb{D} \leftarrow \mathbb{D} - e$;
11: **if** $|\mathbb{D}| < minSup$;
12: **break**;

Subprocedure 1 Subgraph_Mining($\mathbb{D}$, $\mathbb{S}$, s).

1: **if** $s \neq min(s)$
2: **return**;
3: $\mathbb{S} \leftarrow \mathbb{S} \cup \{s\}$;
4: enumerate s in each graph in $\mathbb{D}$ and count its children;
5: **for each** c, c is s' child **do**
6: **if** $support(c) \geq minSup$
7: $s \leftarrow c$;
8: Subgraph_Mining($\mathbb{D}_s$, $\mathbb{S}$, s);

図 13 gSpan の疑似コード(Yan and Han, 2002)

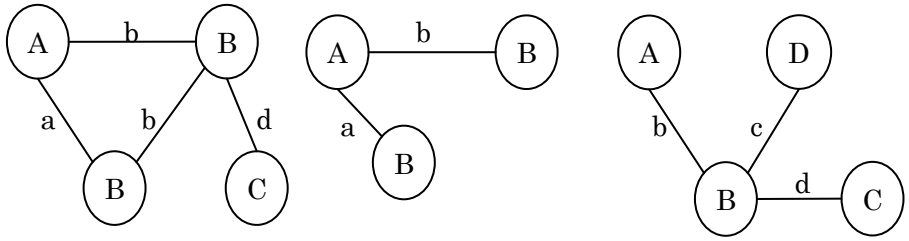


図 14 Frequent connected subgraph 抽出対象となる graph のセット

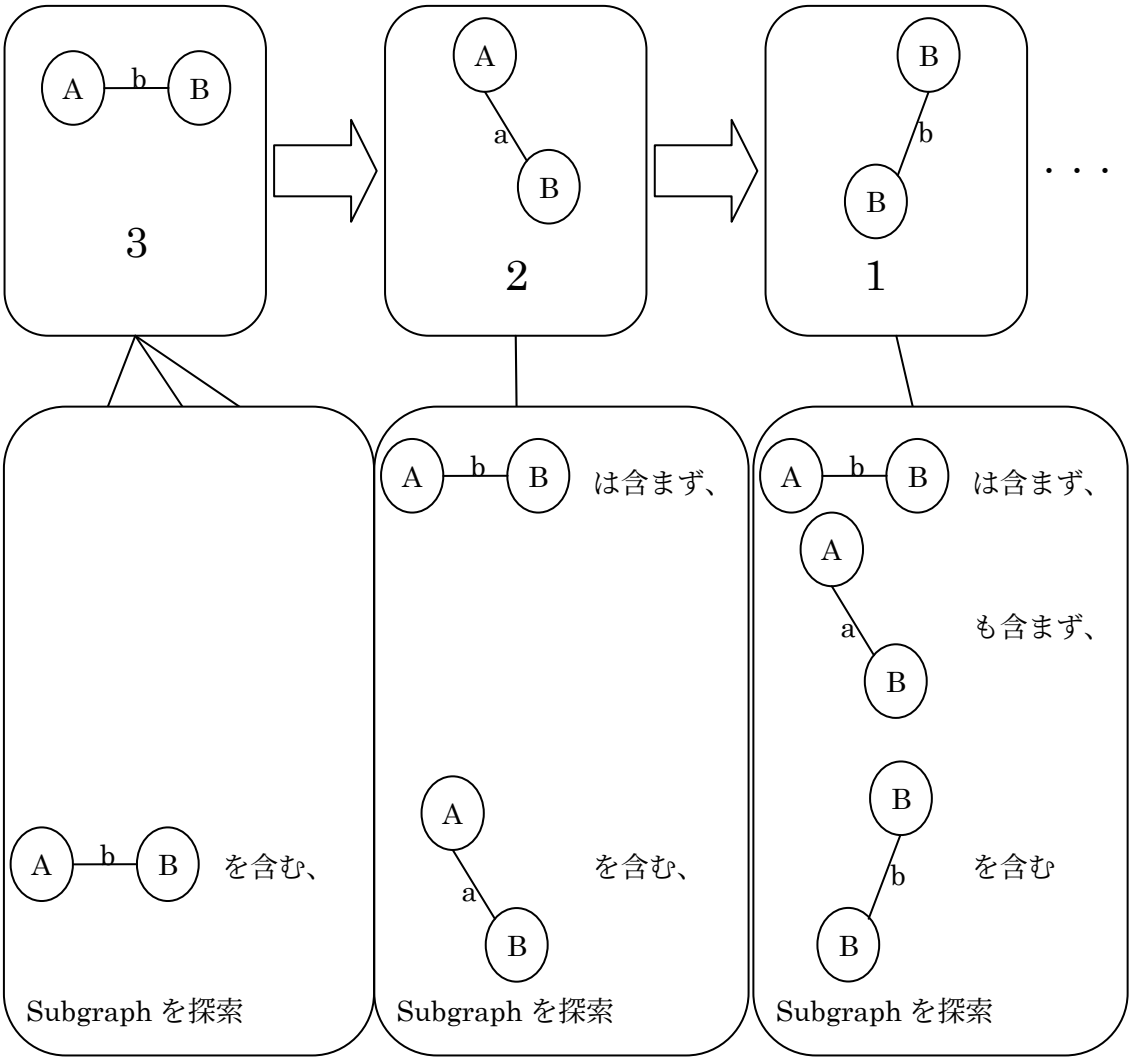


図 15 無駄な探索の回避「開始 edge に着目」

gSpan では各 edge を初期 Subgraph とし、隣接要素を追加する作業を繰り返すことで Connected subgraph を生成している。一度初期 Subgraph に採用された edge は graph から除外することで、別の edge を初期 Subgraph として Connected subgraph の生成を行う際に重複 Connected subgraph が生成されることを回避している。

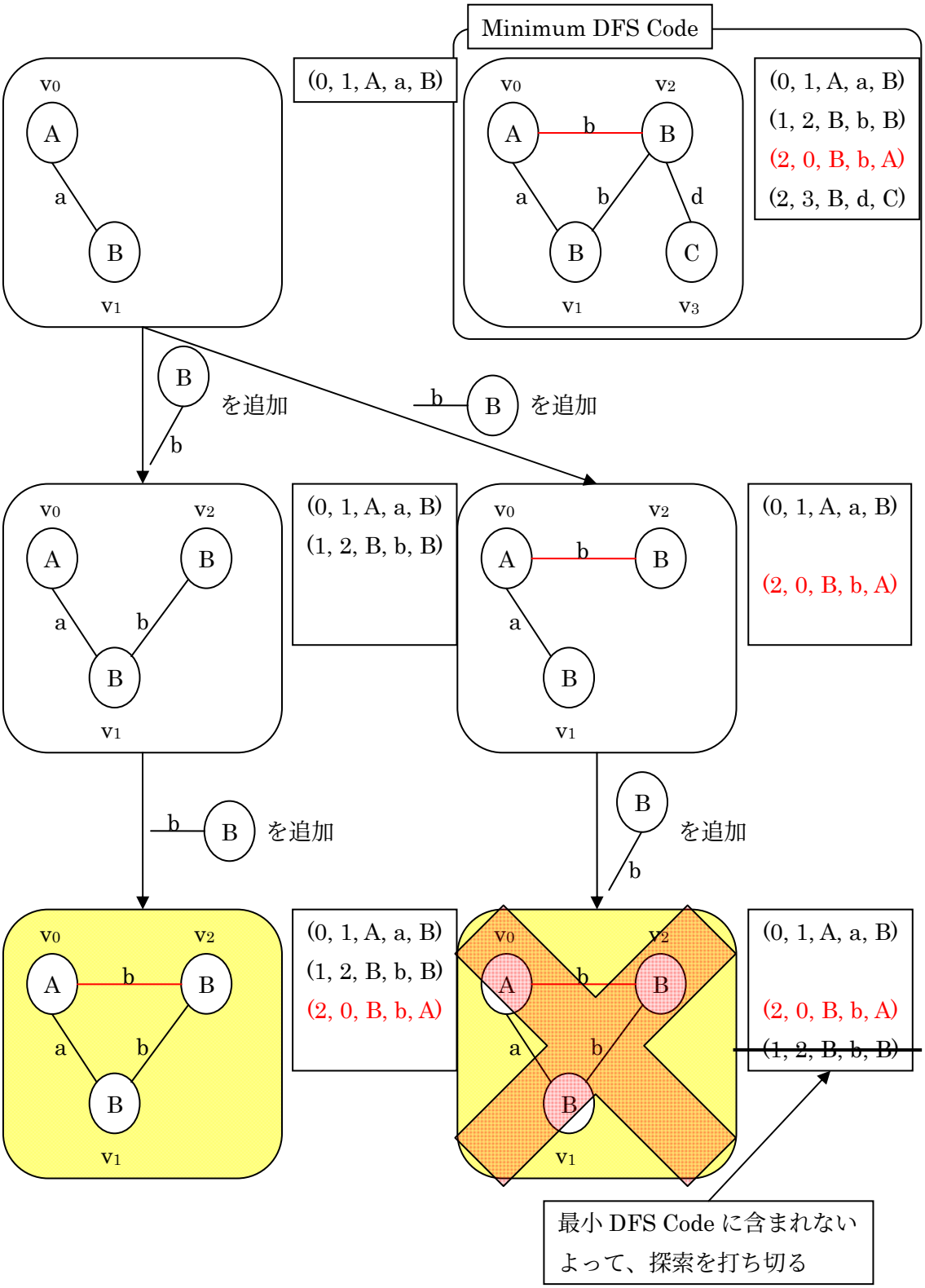


図 16 無駄な探索の回避「Minimum DFS Code との比較」

gSpan では DEF Code が最小 DFS Code の一部でない場合は探索を打ち切ることで無駄な探索を省く。

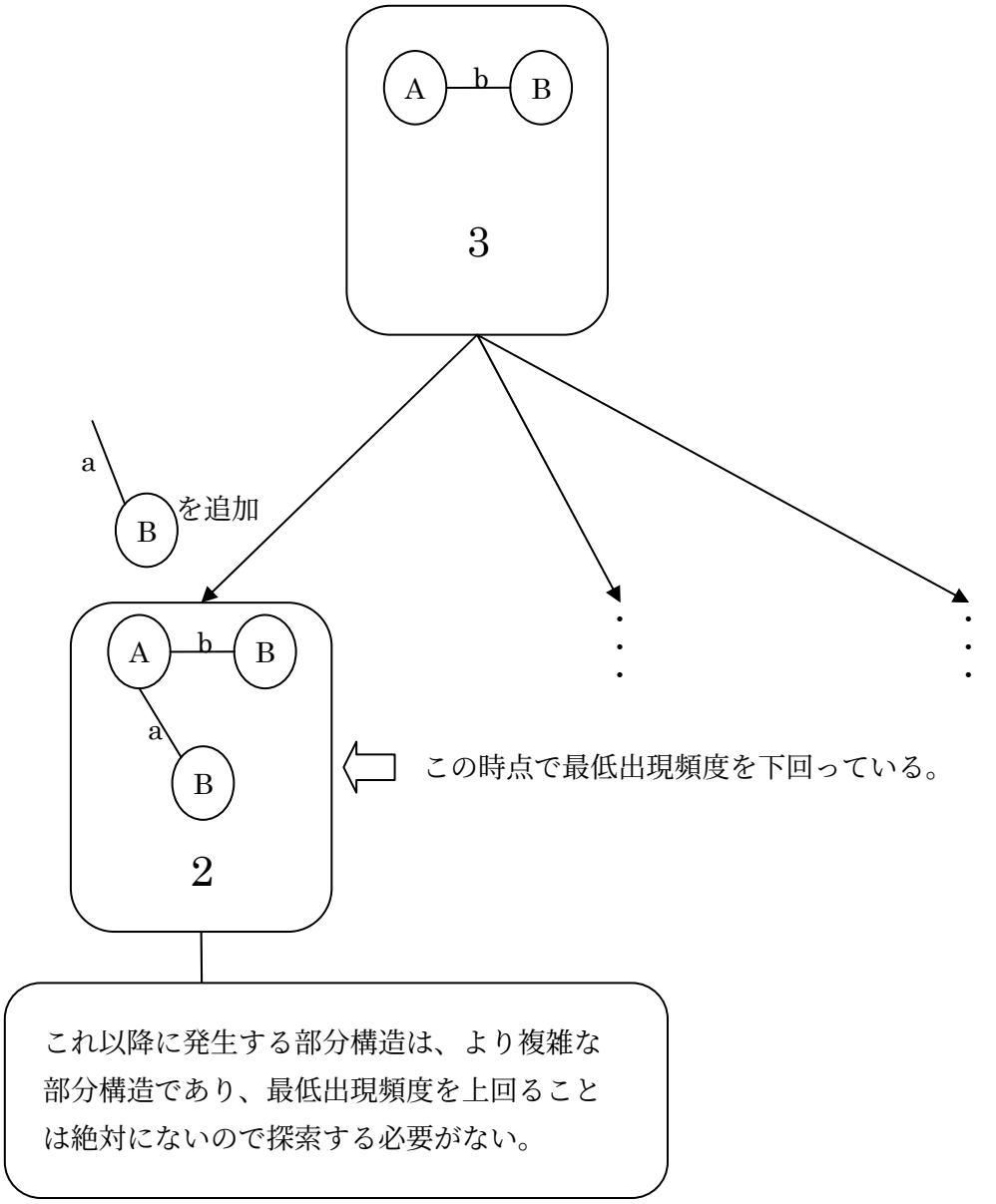


図 17 無駄な探索の回避「出現頻度に着目」

隣接要素を追加することで得た Subgraph が最低出現頻度を下回った場合、それ以降の探索は打ち切ることが可能。このケースでは最低出現頻度を 3 と設定した場合、出現頻度が 2 の Subgraph に到達した段階で、探索を打ち切ることが出来る。

第3章　提案手法の着眼点について

本章では、提案手法の着眼点や概念について説明を行う。アルゴリズムの詳細に関する説明は第 4 章にて行う。まず、第 1.2.2 項で述べた Maximum Common Subgraph から得られる類似係数を用いた代謝経路予測は、Maximum Common Subgraph の取得が NP 困難である為、計算コストが非常に高い。また、仮に全ての計算が完了したとしても、以下に示す理由から正確な予測を行うのは難しい。まず、Tanimoto 係数は以下の式で計算できる。

$$\text{Tanimoto 係数} = \frac{\text{共通部分の大きさ}}{\text{共通部分の大きさ} + \text{相違部分の大きさ}}$$

これは、仮に同じタイプの反応、つまり相違部分が等しい反応であっても、共通部分の大きな化合物間の Tanimoto 係数は大きな値を取り、共通部分の小さな化合物間の Tanimoto 係数は小さな値を取ってしまうことを意味する。例えば、図 18 のケースにおいては左右の反応はどちらも隣り合う 2 つのヒドロキシル基から環を構成する同じタイプの反応であるが、Tanimoto 係数は一致しない。

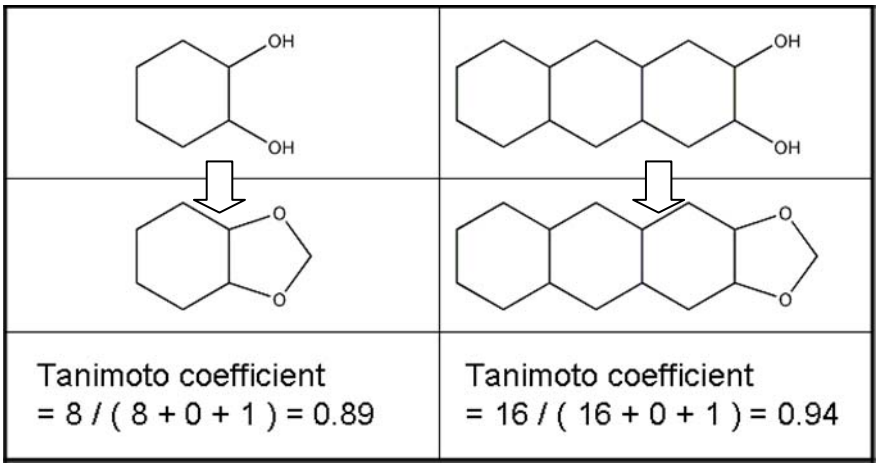


図 18 代謝経路予測に Tanimoto 係数を用いる場合の問題点
右の反応では反応の前後での共通部分は六員環の炭素 6 個と酸素 2 個、相違部分は反応後の酸素間の炭素 1 個であり（水素は無視している）、Tanimoto 係数は 8/(8+0+1)=0.89 となる。左の反応では反応の前後での共通部分は六員環が 3 つ連なっている部分の炭素 14 個と酸素 2 個、相違部分は反応後の酸素間の炭素 1 個であり、Tanimoto 係数は 16/(16+0+1)=0.94 となる。左右の反応はどちらも 2 つのヒドロキシル基から環が生成される同じ反応であるが、Tanimoto 係数では同じ値を出力してくれない。

Tanimoto 係数を用いた場合の問題点を以下の例により示す。図 19 は原子と結合が一つずつ増えていく鎖状の化学構造からなるデータセットである。図 19 のデータセットに対し炭素の伸長反応を仮定する(図 20)。これに対して、Tanimoto 係数を用いて経路予測を行った場合にどうなるのかを図 21、図 22、および図 23 に示した。Tanimoto 係数の計算方法では、同じタイプの反応であっても共通部分の大きさに依存して値が変化する為、閾値を小さく設定した場合には構造が複雑な化合物間において False-Positive が増加し(図 21、図 22)、閾値を大きく設定した場合には構造が単純な化合物間において False-Negative が増加してしまう(図 22 図 23)。

定量的構造活性相関を行うのであれば、共通部分の占める割合の小さい A と B の間の活性の違いは大きく、共通部分の占める割合が大きい E と F の間の活性の違いは小さいと考えるのは妥当であるが、代謝経路予測を目的として基質と生成物間の類似係数を導入するのであれば、少なくとも相違部分の変化にのみ着目した類似係数（例、Manhattan 係数）を用いるべきである。

また、類似係数を用いること自体も問題を含んでいる。一度の代謝反応で起こる変化の度合いはメチル化(-CH3 が付加する反応)やヒドロキシル化(-OH が付加する反応)のように小さな構造変化しか伴わないケースもあれば、糖付加や多量体を構成する反応のように基質と生成物間での変化の度合いが大きな反応も存在する。このことは代謝反応によって基質と生成物間の類似度は様々な値を取ることを意味しており、代謝物間の類似度を計算した後、特定の閾値を設定することで代謝経路を予測する方法では予測結果が曖昧になってしまうことを意味する。

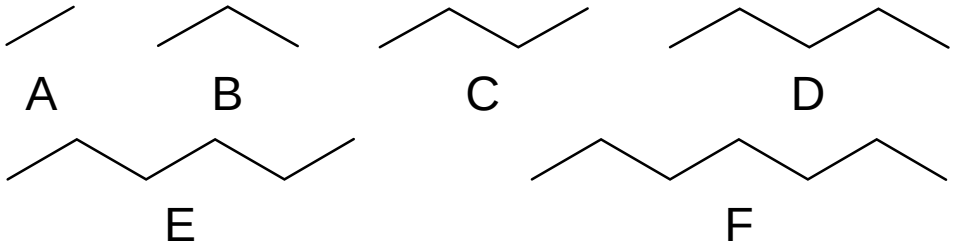


図 19 鎖状構造のみからなる化学構造のセット

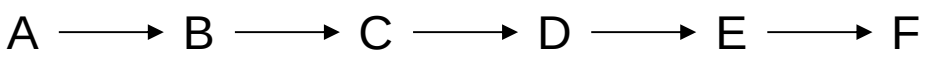


図 20 望ましい代謝経路予測結果

表 1 Maximum Common subgraph から求めた化合物間の Tanimoto 係数

	A	B	C	D	E	F
A	1.00					
B	0.67	1.00				
C	0.50	0.75	1.00			
D	0.40	0.60	0.80	1.00		
E	0.33	0.50	0.67	0.83	1.00	
F	0.29	0.43	0.57	0.71	0.86	1.00

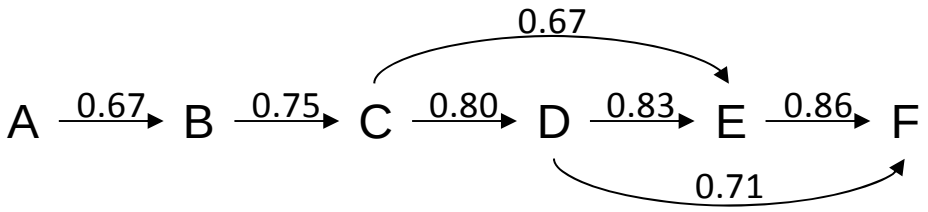


図 21 Tanimoto 係数の閾値を 0.67 として経路予測を行った場合
C→E および D→F が誤った予測として検出される(False-Positive が増加)

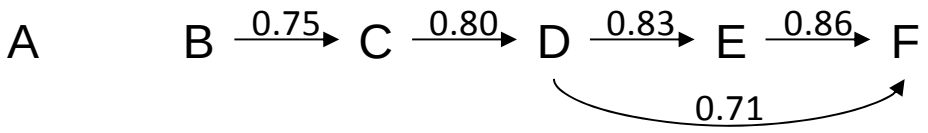


図 22 Tanimoto 係数の閾値を 0.71 として経路予測を行った場合
A→B が予測できず(False-Negative が増加)、D→F が誤った予測として検出される(False-Positive が増加)。

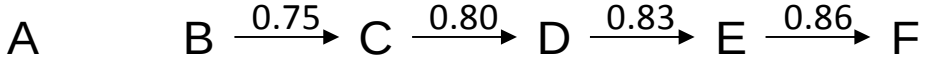


図 23 Tanimoto 係数の閾値を 0.75 として経路予測を行った場合
A→B が予測できない(False-Negative が増加)。

そこで、本研究では化学構造のペアを代謝経路として採用するかどうかを決定する際に、類似係数による数値比較を行う代わりに、Graph-based approach による比較を行うことで予測精度の向上を試みた。先行研究のアプローチが、類似度が閾値以上の化学構造ペアを代謝経路として予測するのに対し、提案手法では、構造が最も近い化学構造を代謝経路として予測する。ここで、「構造が最も近い化学構造」とは第 2.1 節で定義を行った Parent subgraph と Child subgraph の関係にある化学構造を指す。例えば、図 19 の場合では、6 つの化学構造は A ⊂ B ⊂ C ⊂ D ⊂ E ⊂ F の関係にある。この時、Parent-Child の条件を満たす包含関係は、A ⊂ B、B ⊂ C 、C ⊂ D、D ⊂ E 、E ⊂ F となり、図 20 の代謝経路に相当する。ここで、図 20 の代謝反応は全て付加反応に相当する。付加反応や脱離反応は、基質と生成物のどちらか片方の化学構造がもう片方の化学構造を完全に包含するので Parent-Child の表現が可能であるが(図 24)、置換反応では基質と生成物間に包含関係が存在しない為、Parent-Child の関係を得ることが出来ない(図 25)。そこで、Maximum Common Connected subgraph (以下 MCCS と呼ぶ)を中間構造として導入することにより「基質 ⊃ 中間構造、中間構造 ⊂ 生成物」の形式で包含関係を表現することで、置換反応も考慮に入れた包含関係に着目した代謝経路予測を可能とした(図 26)。

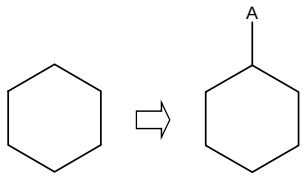


図 24 付加反応の例
付加反応や脱離反応では片方の化学構造がもう片方の化学構造を完全に包含する。

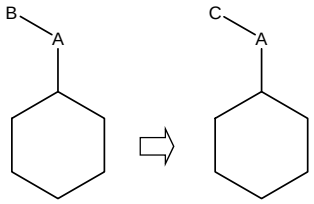


図 25 置換反応の例
置換反応では、両者の化学構造間に包含関係が存在しない。

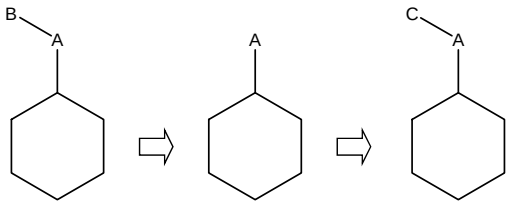


図 26 Maximum Common Connected subgraph を用いた置換反応の表現
置換反応前後の化学構造の Parent subgraph として、Maximum Common Connected subgraph を導入することで包含関係に着目した代謝経路予測が可能になる。

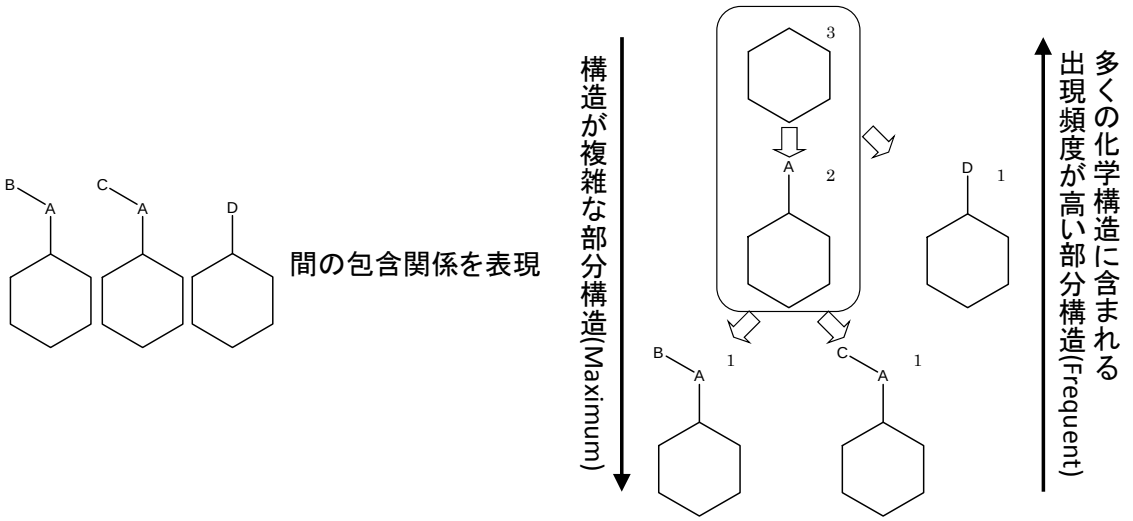


図 27 Maximum Frequent Connected subgraph を用いた複数化学構造間の代謝経路予測
右肩の数字はいくつの化学構造に含まれるか(出現頻度)を表している。図中枠に囲まれた MFCS は中間構造として抽出した MFCS である。ここで、MFCS には入力とした Chemical graph と同形な Subgraph が必ず含まれる点に注意したい、図の例では入力とした 3 つの化学構造は出現頻度が 1 の MFCS として取得される。

3 つ以上の化学構造に対して、包含関係に着目した代謝経路予測を行う場合は、MCCS の代わりに Maximum Frequent Connected subgraph(以下 MFCS と呼ぶ)を中間構造として導入することで、置換反応を表現する(図 27)。「Common」と「Frequent」の違いは、対象が 2 つの化学構造の場合、「2 つの化学構造が共通に持つ部分構造」と言う意味で「Common」が用いられるが、対象の化学構造が 3 個以上になった場合、例えば 10 個の化学構造が対象のケースでは、「10 個中 10 個の化学構造が共通に持つ部分構造」以外に「10 個中 7 個の化学構造が共通に持つ部分構造」も意味のある共通部分構造となる。この時、一部の化学構造が共通に持つ部分構造も対象にすることを明示的にする為に、「Common(共通)」という表現の代わりに「Frequent(頻出)」と言う表現を用いる。

MFCS の対象となるのは、入力とした Chemical graph から取得可能な全ての Connected subgraph の内、自分以外の全ての Subgraph に対し「構造が複雑な Subgraph である(Maximum)」もしくは「より多くの化学構造に含まれる、出現頻度が高い Subgraph である(Frequent)」のどちらかの条件を満たす Connected subgraph が相当する。この時、MFCS には入力とした Chemical graph と同形な Subgraph が必ず含まれる点に注意したい。図 27 の例では、入力とした 3 つの Chemical graph と同形な Subgraph が、出現頻度が 1 の MFCS として取得される。よって、取得した MFCS に対して代謝経路予測を行うことで Compound レベルの代謝経路予測が実現できる。

ここで、MFCS を取得するアルゴリズムは、Maximum Common subgraph を取得する

アルゴリズムと同様に、膨大な数の Subgraph の中から目的の Subgraph を取得する NP 困難なアルゴリズムになる。gSpan のように Connected subgraph のみを対象とする探索手法を用いても、複雑な Framework を保有する Chemical graph からは大量の Connected subgraph が取得可能な為、膨大な計算コストが要求される。そこで、計算コストの低減を目的として、「既知の代謝経路上で隣り合う代謝物の Framework 間には包含関係が認められる」特徴に着目し、「Framework を含む Connected Subgraph (Framework-based Connected subgraph)のみを対象として MFCS 抽出を行う」というヒューリスティックを MFCS 抽出アルゴリズムに導入することで、計算コストの大幅な削減を実現した。

図 28 に同数の結合で構成される「複雑な Framework を持つ Chemical graph」と「Framework を持たない Chemical graph」から取得可能な 3 種類の Subgraph「全ての Subgraph、Connected subgraph、Framework-based Connected subgraph」の総数を示した。全ての Subgraph の総数は $2^{結合の数}$と膨大である。Connected subgraph に限定することで総数は大幅に減少するが、複雑な Framework を持つ Chemical graph には、なお多くの Subgraph が存在する。対象をさらに Framework-based Connected subgraph に限定することで、Subgraph の数を Side chain に含まれる結合の組合せの数に抑えることが可能になり、特に複雑な Framework を持つ Chemical graph において Subgraph の数を劇的に減少させることが可能になる。この中から MFCS の抽出を行うアルゴリズムを提案することで、数万化合物に対する代謝経路予測を、実時間で処理することを可能にした。

	Portensterol	Thiothece 460
All subgraph	$2^{(\#\ of\ edges)} = 2^{33} = 8{,}589{,}934{,}592$	
Connected subgraph	5,787,190	7,811
Framework-based Connected subgraph	2×2×2×2×27=432	7,811

図 28 化学構造から取得可能な Subgraph の数

複雑な Framework を持つ化合物(Portensterol)と鎖状構造のみからなる化合物(Thiothece 460)について、Subgraph の数を比較した。どちらの化合物も 33 の結合を持っている（水素との結合は無視している)。制約条件を設けない場合に取得可能な Subgraph の数はどの結合を Subgraph に含むかの組み合わせで決定する為、$2^{結合の数}$個存在する。Connected subgraph に着目することで Subgraph の数は減少するが、それでも複雑な Framework を含むケースでは膨大な数の Subgraph が存在する。Framework-based Connected subgraph に着目すると、Subgraph のバリエーションが Side chain に含まれる結合の組合せの数になる為、Subgraph の数を大幅に削減できる。

第4章　提案手法

本章では、提案手法の詳細について説明を行う。提案手法では、まず、Framework-based MFCS を抽出する。その後、抽出した Framework-based MFCS の包含関係を計算することで、Parent-Child の関係にある Subgraph 二項関係を取得し、これを代謝経路として予測する。第 4.1 節では、提案手法を段階的に説明する為に 3 種類の Subgraph(Framework, MFCS, Framework-based MFCS)の抽出方法について述べる。その後、第 4.2 節において、Subgraph のセットから包含関係に基づいた Parent-Child Subgraph 二項関係を取得する方法について述べる。MFCS および Framework-based MFCS には、入力とした Chemical graph と同形な Subgraph が必ず含まれる。この為、これらに対する代謝経路予測結果が Compound レベルの代謝経路予測結果となる。本章では、図 29 に示した 7 個の Chemical graph を例に用いて、提案手法の説明を行う。

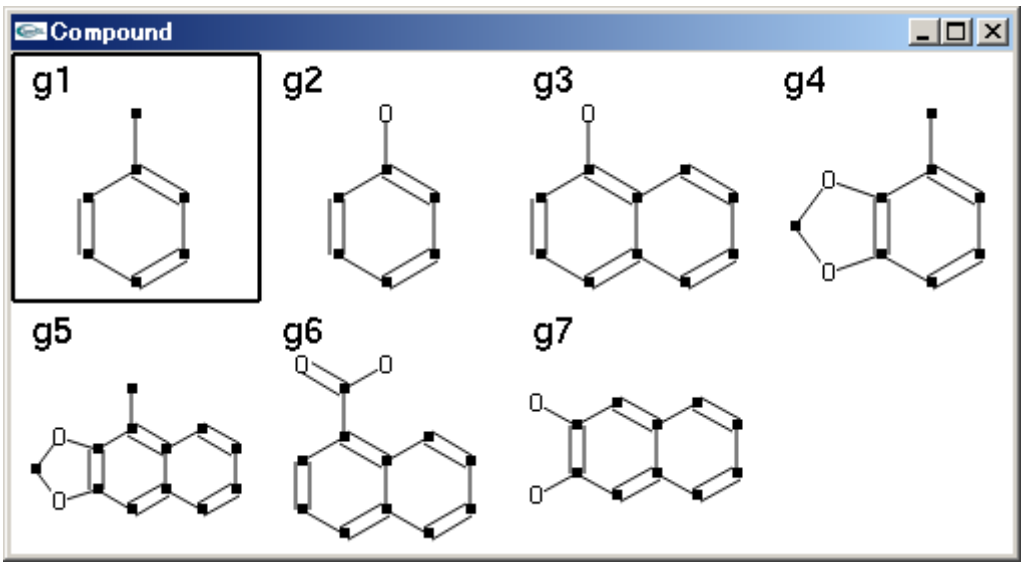


図 29 Subgraph 抽出対象の Chemical graph のセット

第4.1節　　Subgraph の抽出

本節では、以下に述べる 3 種類の Subgraph 抽出方法について説明を行う。第 4.1.1 項では Framework の抽出方法について述べる。提案手法により抽出した Framework は、既知の代謝経路上で隣り合う基質と生成物の Framework 間に包含関係が認められることの検証(第 5.1.1 項)や、Framework-based MFCS を探索する際の初期 Subgraph として用いる。また、Framework に基づいた代謝物の分類は、膨大な数からなる代謝物全体の多様性を理解するのに大いに役立つ。第 4.1.2 項では全ての Subgraph を対象とした MFCS の抽出方法について述べる。MFCS の抽出は NP 困難な問題であり、対象を Framework-based

Connected subgraph に限定しない場合、膨大な計算コストが要求される。第 4.1.3 項では、提案手法で代謝経路予測に用いる Framework-based MFCS の抽出方法について述べる。第 4.1.2 項で説明する MFCS 抽出アルゴリズムの初期 Subgraph に第 4.1.1 項で取得した Framework を用いることで、計算コストの低減及び予測精度の向上を実現した。

第4.1.1項　　Framework

Framework は以下の手順で取得することが出来る。

1. N 個の Chemical graph のセット G(N)を入力とする。
2. 全ての G(N)に対し、次数 1 の node が無くなるまで要素取り除く操作を繰返すことで、Framework を取得する 。
3. 得られた Framework のセットを FG(N)とする。
4. FG(N)からユニークな Framework を抽出し、UFG(M)を得る。

図 29 の 7 個の Chemical graph からは 4 個の Unique Framework が取得できる(図 30)。

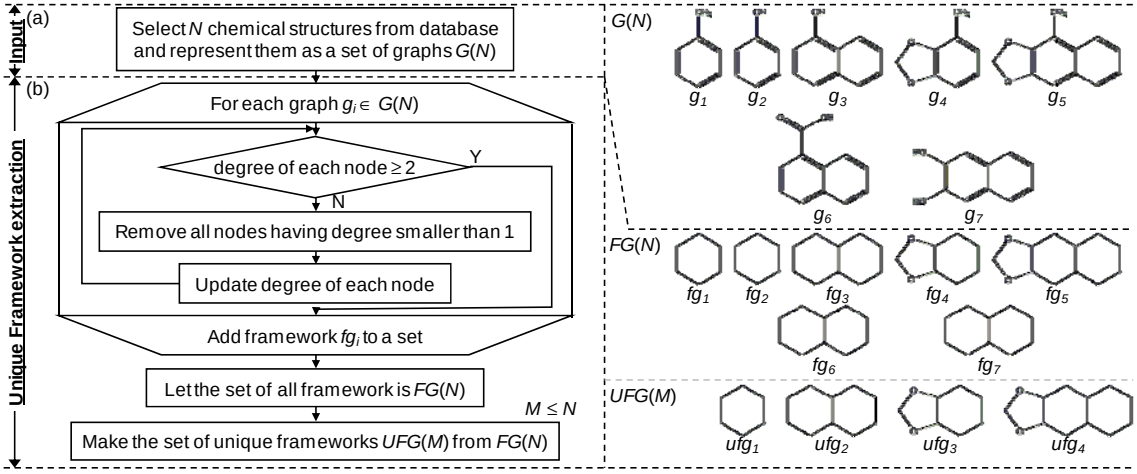


図 30 Unique Framework の抽出

第4.1.2項　　Maximum Frequent Connected subgraph

Maximum Frequent Connected subgraph (MFCS)を取得するアルゴリズムは大きく分けて 2 つの関数、MFCS_Mining_Initialize()および MFCS_Mining()から構成される。本項の最後に疑似コードを示す。Unique Connected subgraph のみを探索するアルゴリズムをベースに、各 Subgraph が MFCS であるかどうかを判定する為の処理、計算コストを低減させる為の処理が組み込まれている。

MFCS 抽出処理のおおまかな流れについて

MFCS_Mining_Initialize()は、MFCS 取得対象とする Chemical graph のセットを入力とし、初期設定を行った後、Chemical graph に含まれる各 node を初期 Subgraph として、MFCS_Mining()を呼び出す。MFCS_Mining()では、入力された Subgraph がいくつの Chemical graph に含まれるか(出現頻度(Support))を計算する。Support が最低出現頻度(MinSup)以下であった場合、隣接要素を追加することで取得可能な Child subgraph の Support も必ず MinSup 以下になるので探索を打ち切る(第 2.2 節[定理 2])。Support ≥ MinSupの場合は、探索中の Subgraph に隣接要素を追加することで Child subgraph を生成し、生成した Child subgraph を入力として MFCS_Mining()を再帰的に呼び出す。

Unique Connected subgraph 抽出処理

この時、隣接要素を追加する順番が異なるだけの重複 Connected subgraph が大量に生成される(図 31)。探索の過程で重複のない Unique Connected subgraph のみに対して Frequent Connected subgraph 判定処理を実行する為に、gSpan では初期 edge に対する排他的処理(図 15)および Minimum DFS Code との比較処理(図 16)を行うことで探索の打ち切りを行っていたが、提案手法では以下に述べる方法を用いて同一の問題を解決した。

まず、提案手法は初期 Subgraph として node を利用している。gSpan において初期 edge に対して行われていた排他的な処理は、初期 node に対して行われる。図 31 の例では、まず node A を初期 Subgraph とする探索で、node A を含む全ての Connected subgraph の探索が完了する。よって、node B や node C を初期 Subgraph とする探索を行う際に、node A を含む Connected subgraph を除外することで、重複 Connected subgraph の一部を探索木上から排除できる(図 32)。同様に node B を初期 Subgraph とする探索で、(node A は含まず)node B を含む全ての Connected subgraph が出現するので、node C を初期 Subgraph として探索を行う際に、node A と node B を含む Connected subgraph を除外することで、初期 node に関連する重複 Connected subgraph の発生を回避することが出来る(図 33)。MFCS_Mining()の内部では探索中の Subgraph に edge(および edge に接続している node)を追加することで Child subgraph の生成を行っている。ここで、追加可能な隣接 edge が複数存在した場合、edge の数だけ Child subgraph は生成可能である。探索中の Subgraph A に追加可能な edge A, edge B, edge C が存在した場合、まず Subgraph A に edge A を追加した Child subgraph A を入力として MFCS_Mining()が再帰呼び出しされる。この探索で Subgraph A と edge A を含む全ての Connected subgraph は生成されるので、別の edge B や edge C を追加することで得られる Child subgraph を入力とした探索では、edge A を含む Connected subgraph を除外して探索を行う。edge B を追加して得た Child subgraph B に関する探索が完了した後、edge C を追加して得た Child subgraph C を用いた探索では、edge A と edge B は追加しないように探索を行うことで、重複 Connected subgraph の生成を回避できる(図 34、図 35)。

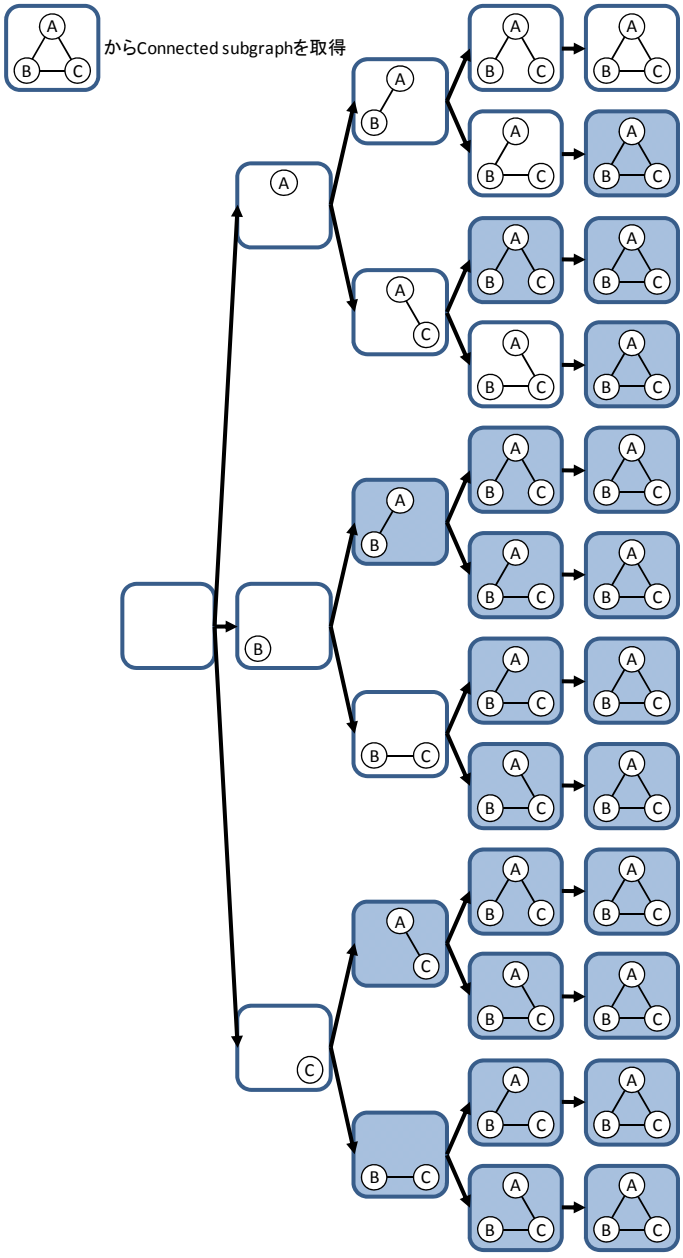


図 31 隣接要素を追加することで得られる Connected subgraph

入力した graph に含まれる各 node を初期 Subgraph として、隣接要素を追加する作業を繰り返すことで、Connected subgraph を取得することが出来るが、隣接要素を追加する順番が異なるだけの重複 Connected subgraph(背景色が付いている Subgraph)が大量に生成されてしまう。これらの Subgraph が生成されないような工夫が必要である。

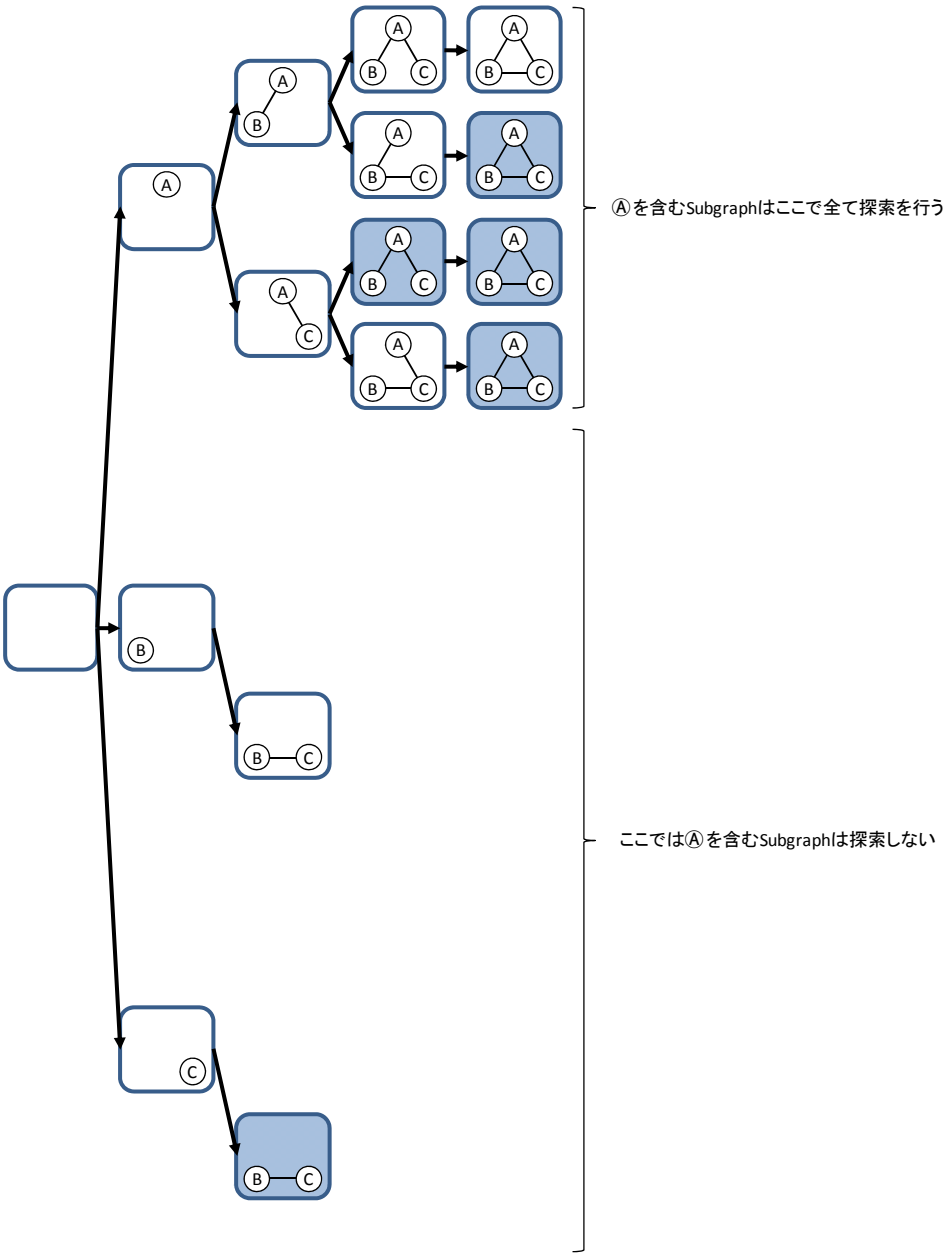


図 32 Unique Connected subgraph の取得 1(初期 node に着目)

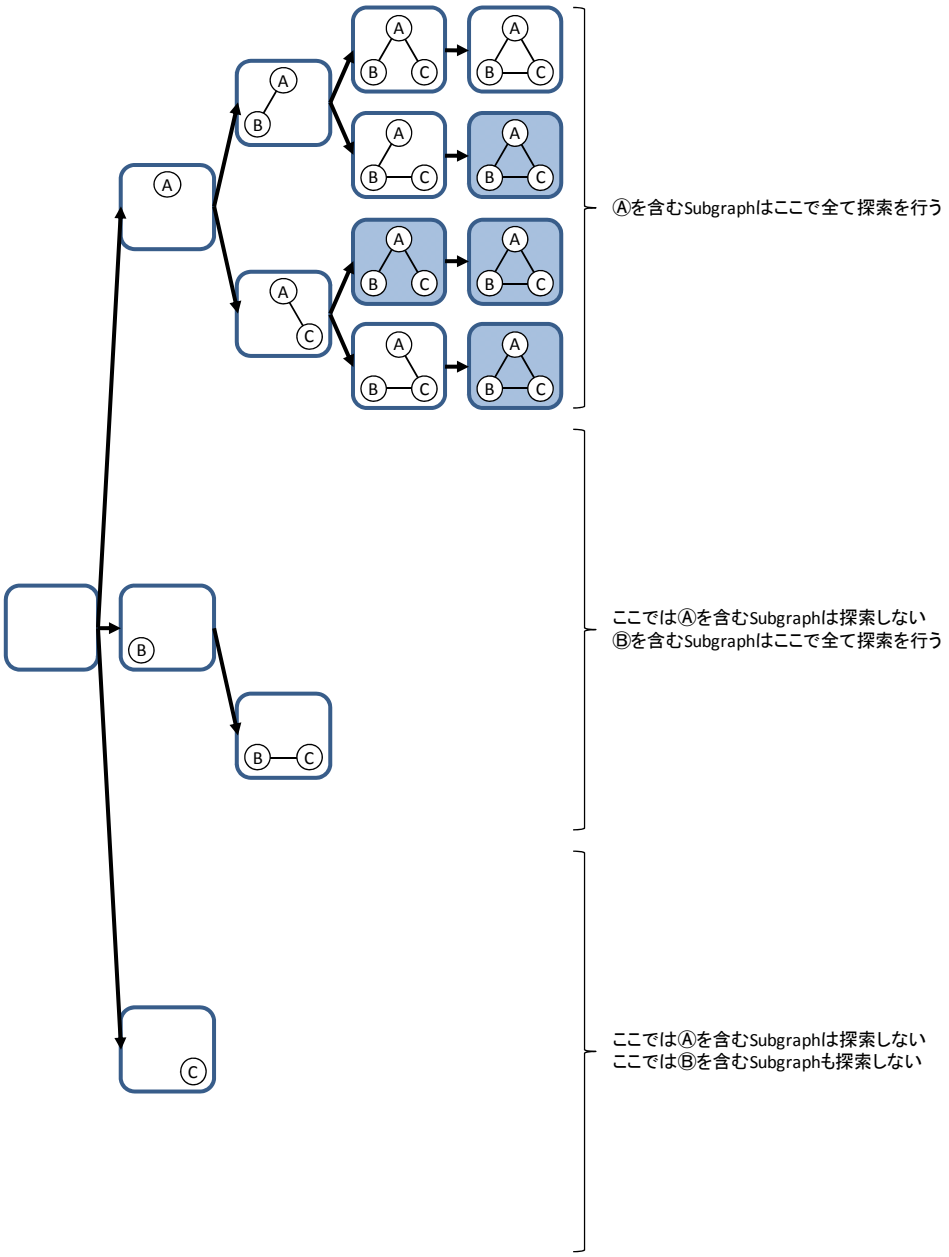


図 33 Unique Connected subgraph の取得 2(初期 node に着目)

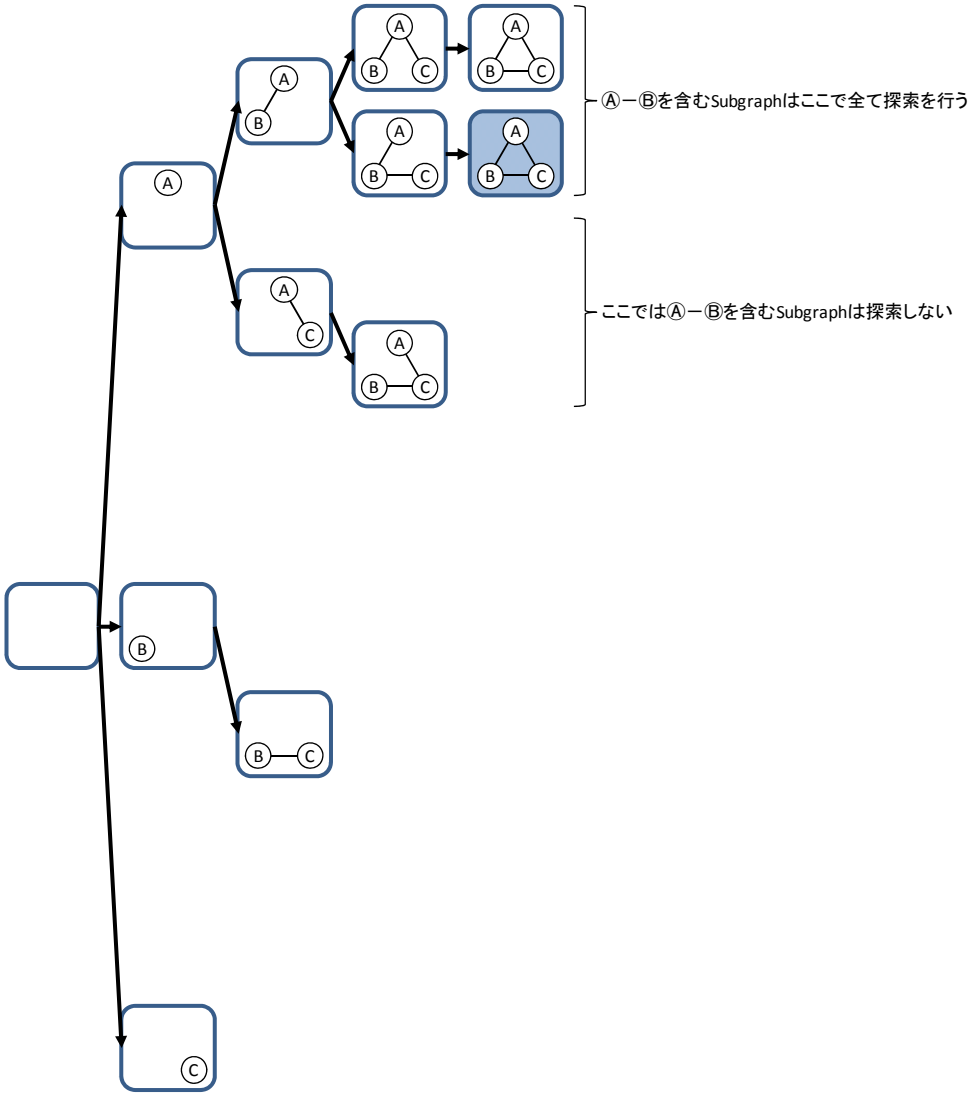


図 34 Unique Connected subgraph の取得 3(Child subgraph 生成時の追加 edge に着目)

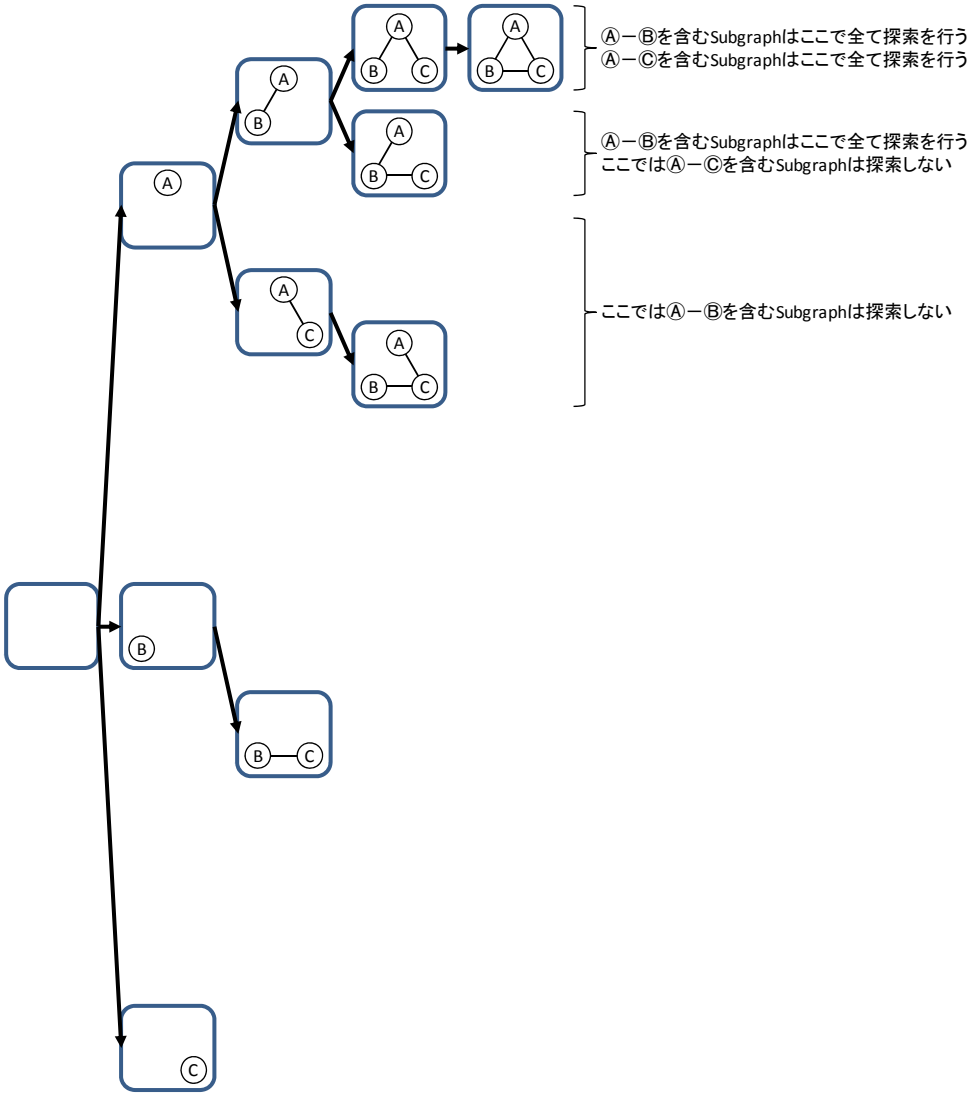


図 35 Unique Connected subgraph の取得 4(Child subgraph 生成時の追加 edge に着目)

Unique Connected subgraph 抽出処理の実装

Unique Connected subgraph 抽出処理の実装は、2 つの配列 IgnoreNode[ ]および IgnoreEdge[ ]を用いることにより達成される。それぞれ Subgraph 抽出元の Chemical graph の node 配列および edge 配列と対応しており各要素の状態が格納される。

IgnoreNode には、対応する node の状態に応じて以下の値が格納される。[状態 1] 探索中の Subgraph に含まれている場合、IgnoreNode には Subgraph に追加された時の再帰呼び出しの深さ(Depth)が格納される。[状態 2] 探索中の Subgraph には含まれないが、この後の探索で追加可能である場合、IgnoreNode = 0 が格納される。[状態 3] 探索中の Subgraph に含まれず追加もできない場合、IgnoreNode = -1 が格納される。

MFCS_Mining_Initialize()の 20 行目で QueryGraph に含まれる node A を初期

Subgraph に設定し、21 行目で node A が Depth=1 で Subgraph に追加されたことを記録する。23 行目で呼び出す MFCS_Mining()によって node A を含む全ての Subgraph を対象に MFCS の抽出を行う。その後、別の node を初期 Subgraph に設定し、23 行目の MFCS_Mining()を呼び出す際には、node A を含む全ての Subgraph は重複 Subgraph となる為、探索から除外する必要がある。そこで、MFCS_Mining_Initialize()の 24 行目で探索済みの node に対応する IgnoreNode に-1 を設定し、MFCS_Mining()の 21～54 行目において探索中の Subgraph に edge を追加して Child subgraph を生成する際、29,30 行目の判定処理で IgnoreNode に-1 が設定された node に繋がっている edge を追加対象から除外することで無駄な探索を回避する。

IgnoreEdge には、対応する edge の状態に応じて以下の値が格納される。[状態 1] 探索中の Subgraph に含まれている場合、IgnoreEdge には Subgraph に追加された時の再帰呼び出しの深さ(Depth)が格納される。[状態 2] 探索中の Subgraph には含まれないが、この後の探索で追加可能である場合、IgnoreEdge = 0 が格納される。[状態 3] 探索中の Subgraph に含まれず追加もできない場合、IgnoreEdge には Subgraph に追加してはいけないことが確定した際の再帰呼び出しの深さ(Depth)が格納される。ここで、[状態 1]と[状態 3]は共に Depth が格納される。これは、「Subgraph に追加してはいけない edge」とは「Subgraph に追加済みの edge」もしくは「Subgraph には含まれないが追加してはいけない edge」の 2 種類が存在する為で、両者を区別することなく一度の判定処理で済ませることで計算コストを低減させることを目的とした結果である。

MFCS_Mining()の 31 行目で探索中の Subgraph A に edge B を追加することで新しい Subgraph C(Child Subgraph)を生成し、32 行目で edge B は Depth + 1 で追加されたことを記録する。40 行目で再帰呼び出しされる MFCS_Mining()は Subgraph A と edge B を含む全ての Subgraph(図 36①)を対象に MFCS の抽出を行う。その後、43 行目において edge B は Subgraph C から取り除かれ、Subgraph C は Subgraph A に戻る。別の edge を Subgraph A に追加して 40 行目の MFCS_Mining()を呼び出す際には、Subgraph A は含むが edge B を含まない Subgraph(図 36②)のみを対象に MFCS の抽出を行う。28 行目の処理において IgnoreEdge が 1 以上の edge は含まないようにすることで、図 36②を探索している過程で図 36①が再度出現することを防止し、重複 Subgraph の生成を回避する。21～54 行目の for ループを抜けた段階で、Subgraph A を含む全ての Subgraph(図 36①②)に対する抽出処置は完了するが、その他の Subgraph(図 36③④)に関しては抽出処理が完了していない。図 36③に該当する Subgraph は Edge B を含む Subgraph も存在する為、55～57 行目で Depth + 1 で取り除かれた Edge に対応する IgnoreEdge を 0 に戻す。

IgnoreNode と IgnoreEdge を用いた以上の処理により、重複 Subgraph の生成を防ぎ、計算コストを低減させている。gSpan では、同様の効果を Minimum DFS Code を用いて実現しているが、gSpan の手法では全ての Subgraph に対して DFS Code が Minimum DFS Code から取得可能かどうかを調べる処理が発生する。また、一度重複 Subgraph を出現さ

せた後に Minimum DFS Code との比較で重複 Subgraph を除外している。提案手法では重複 Subgraph が出現しないので、計算コストがより低くなる。

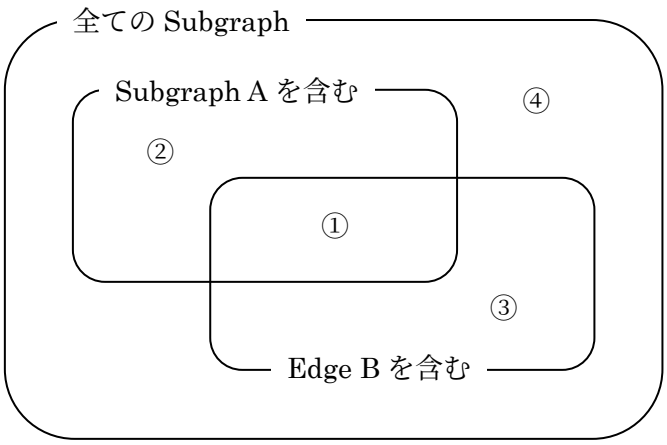


図 36 IgnoreEdge を用いた排他的処理の概念図

MFCS 判定処理

ここまでの処理で MFCS の対象となる Subgraph を Unique Connected subgraph に限定することが出来る。探索中の Subgraph が MFCS であるかどうかの判定は、全ての Child subgraph を入力とした MFCS の探索が完了した後、探索中の Subgraph と Support が等しい Child subgraph が一つも存在しなかった場合、MFCS の候補とする。MFCS であることが確定するのではなく、候補となる理由は、Unique Connected subgraph を抽出する為に除外した重複 Connected subgraph に該当する Child subgraph の Support の計算が行われない為であるが、Child subgraph との比較の代わりに、過去の探索において MFCS であることが確定した Subgraph と包含関係および Support の比較を行うことで、探索中の Subgraph が MFCS であるかどうかを確定することが出来る。

MFCS_Mining()の再帰処理による Unique Connected subgraph の生成過程と各 Subgraph の Support 計算結果を単純な Chemical graph を例に説明する。図 29 の 7 個の Chemical graph から MFCS を抽出する場合、Chemical graph g1 を基にした探索では図 37 に示す Unique Connected subgraph が生成される。ここで、各 Subgraph の右肩の数字は Support を意味している。図 37 の各 Subgraph は、図 38 の順路で探索が行われる。Support の値が等しい Child subgraph が存在する場合、該当する Subgraph は MFCS の候補から除外される(図 39 で赤枠の付いた Subgraph)。その後、探索中の Subgraph と過去の探索で見つかった MFCS の比較の結果、図 40 で青枠の付いた Subgraph が MFCS の候補から除外される。結果として、図 29 の 7 個の Chemical graph から MFCS を抽出する場合、g1 からは図 40 右上の 2 つの Subgraph が MFCS として抽出される。

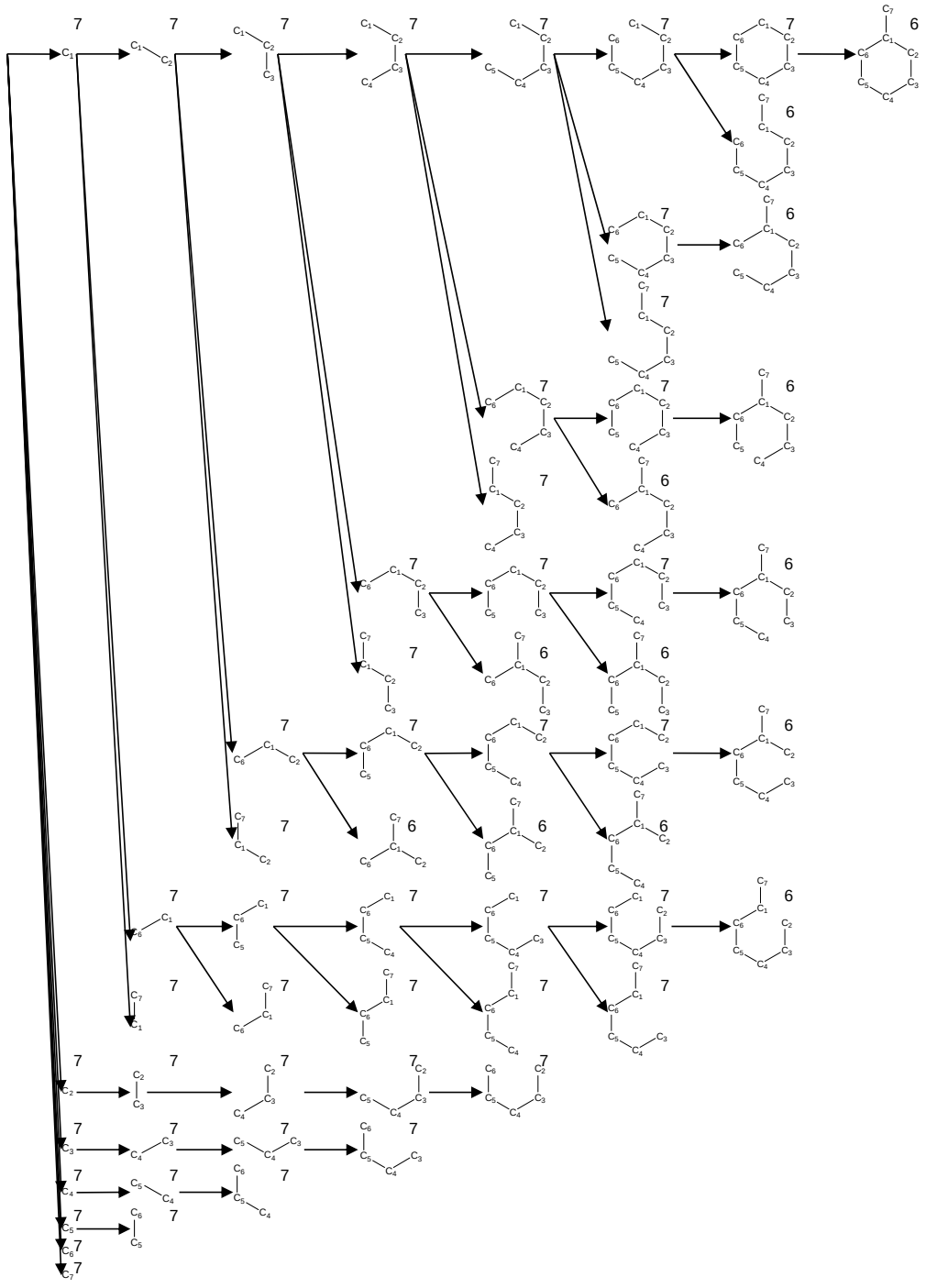


図 37 重複 Subgraph を生成しない探索木

図 29 の Chemical graph g1 から取得可能な Unique Connected subgraph、右肩の数字は出現頻度(Support)を表している。

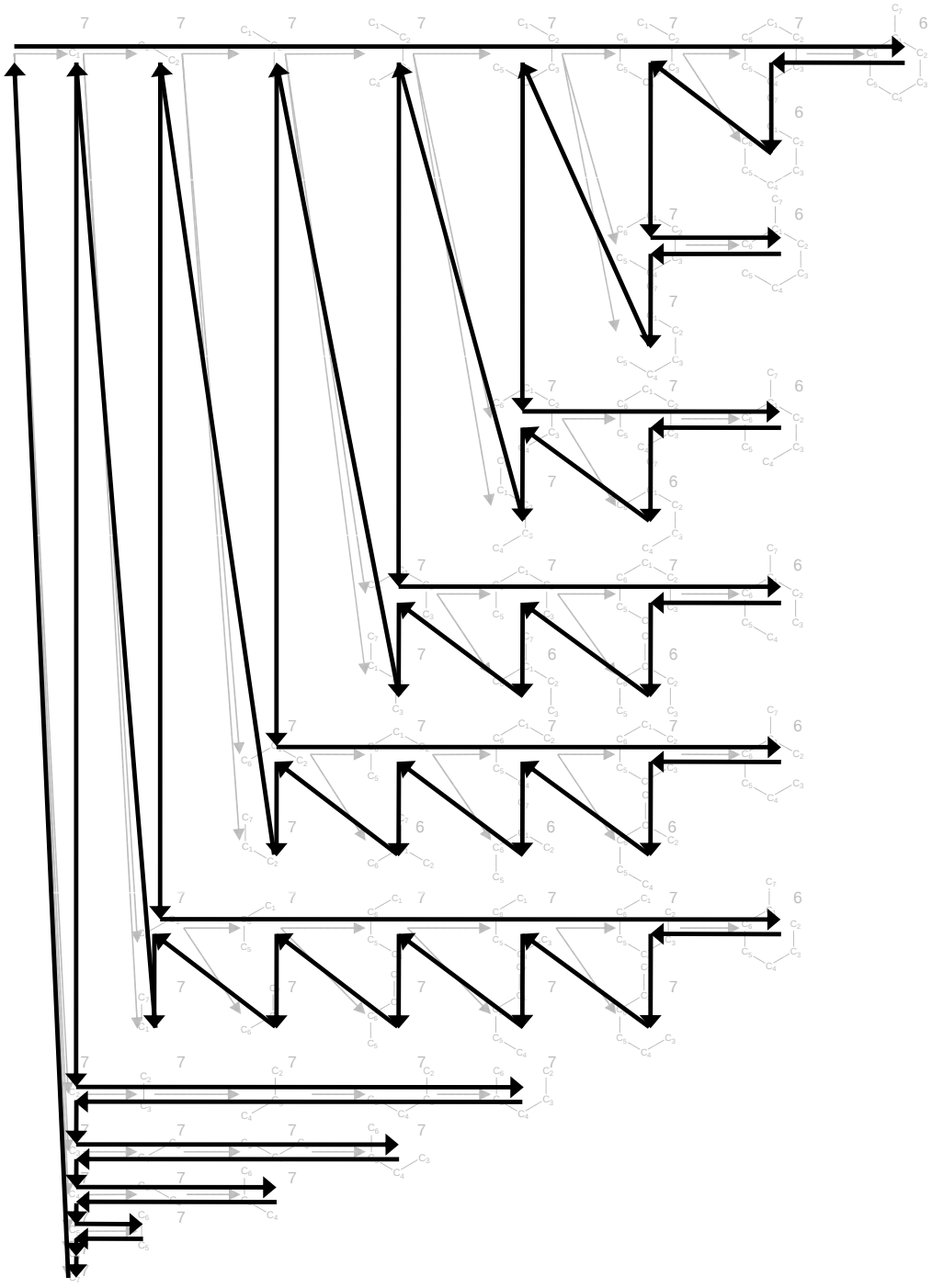


図 38 Connected subgraph 探索順路

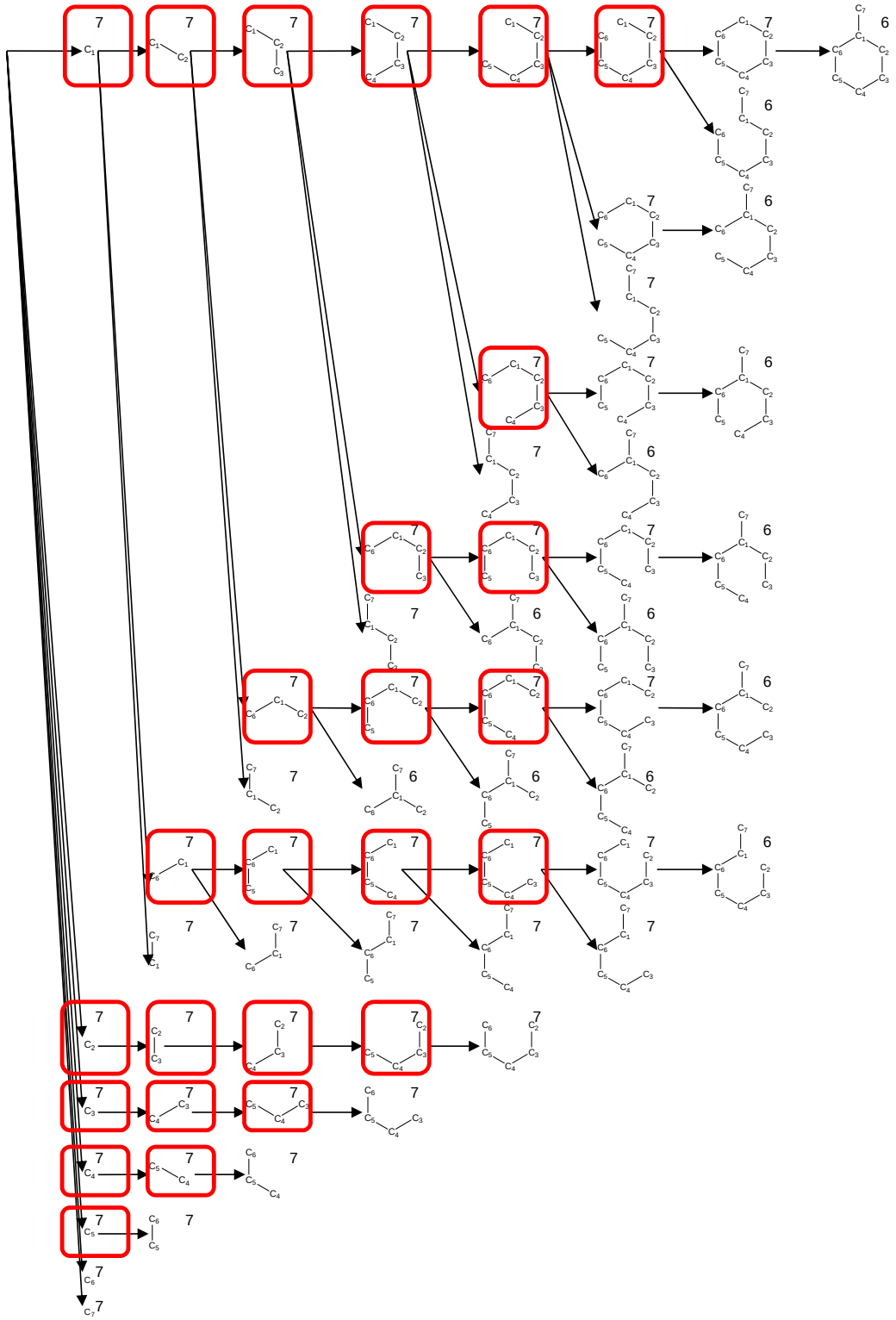


図 39 探索中の Subgraph と Child subgraph の比較

「探索中の Subgraph の Support」＝「Child subgraph の Support」である場合、MFCS の候補から除外する。

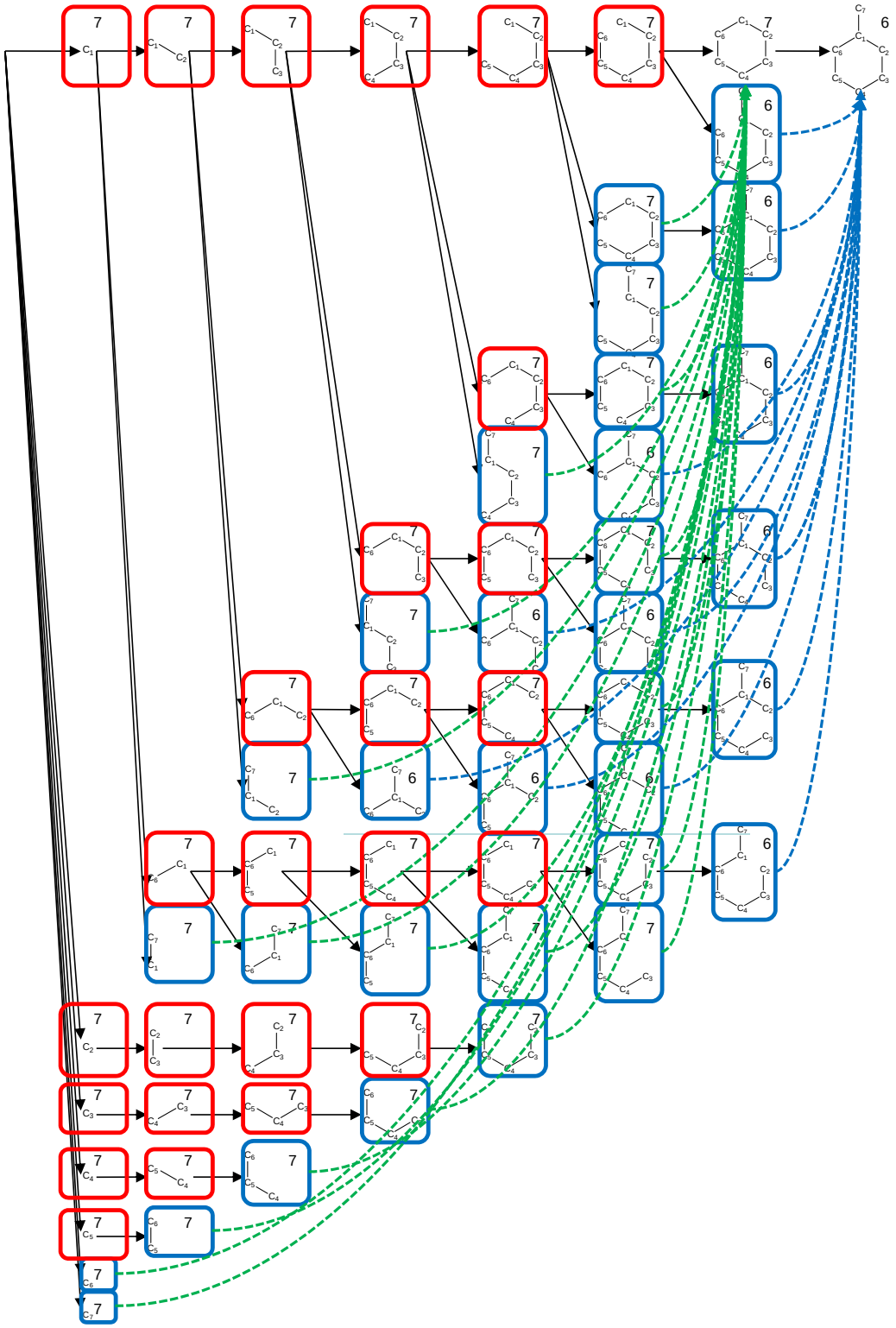


図 40 探索中の Subgraph と過去の探索で見つかった MFCS の比較

MFCS の候補となった Subgraph に対し、過去の探索で見つかった MFCS の中に「探索中の Subgraph」⊂「MFCS」、かつ、「探索中の Subgraph の Support」=「MFCS の Support」を満たす MFCS が一つも存在しなかった場合、探索中の Subgraph を MFCS に追加する。

MFCS 抽出アルゴリズムの高速化

以上の処理で MFCS の取得は可能である。しかし、MFCS_Mining()は再帰関数であり、打ち切り処理を何も入れない場合、全体の処理を通じて Unique Connected Subgraph の数と対応した回数だけ呼び出される。全ての Subgraph を対象にするのと比較して Unique Connected subgraph のみに対象を限定することで大幅な数の Subgraph を探索対象から除外できるが、Chemical graph が複雑なケース、例えば図 28 に示した代謝物 Portensterol の場合、5,787,190 回呼び出されることになり、計算コストが非常に高い。この為、計算コストを低減する為に、さらにいくつかのアルゴリズムを組み込むこととした。

実装の全体的な流れについて

MFCS_Mining()の全体的な流れは、1～12 行目で各入力 Chemical graph との包含関係を確認した後、13～15 行目で Support を計算する。16, 17 行目で Support および制限時間で探索打ち切り処理を実行する。18～57 行目で探索中の Subgraph に隣接要素を追加することで Child subgraph を生成し MFCS_Mining()を再帰呼び出しする。全ての Child subgraph に対する評価が完了した後、58～66 行目において探索中の Subgraph が MFCS に該当するかどうかの評価を行い、MFCS に該当していた場合、67～69 行目で探索中の Subgraph を MFCS に追加する。

メモリ確保回数の低減

ソフトウェアが動作する際、1.メモリ上にデータを格納する領域を確保する。2.データを格納する。3.格納したデータを使って計算を行う。4.いらなくなったデータ領域を解放する。といった一連の処理が実行される。ここで、1 の処理は計算コストが非常に高い。5,787,190 回呼び出される MFCS_Mining()において、その都度データ領域を確保するような処理では、動作速度が非常に遅くなる。この問題を回避する為、MFCS_Mining_Initialize()の 1～3 行目において処理対象データの最大サイズを取得し、4～8 行目において最大サイズで作業用領域を事前に確保し、この後の処理で確保した作業用領域を再利用することで計算コストを低減させる。

複雑な Chemical graph からの MFCS 抽出処理の回避

Subgraph は Chemical graph に含まれる要素の組合せの数だけ存在する。この為、Chemical graph に含まれる要素(node や edge)が増えると Subgraph の数も増え、MFCS の抽出に必要な計算コストも増大する。ここで、Chemical graph の数が n、MinSup(最低出現頻度)が k の場合、n-k+1 個の Chemical graph についてのみ MFCS の抽出を試みればよい。例えば n=10,k=3 の場合、10-3+1=8 個の Chemical graph についてのみ MFCS の抽出を試みればよい。なぜならば、9 個目の Chemical graph から取得した Subgraph A の Support(出現頻度)が 3 であった場合、Subgraph A は 1～8 個目の Chemical graph のどれ

かに必ず含まれる。1～8 個目の Chemical graph から取得可能な Subgraph に関しては、MFCS であるかどうかの判定が全て完了しており、仮に Subgraph A が MFCS であったとしても、出力済みの MFCS のどれかと同形になる。つまり、9 個目以降の Chemical graph からは MFCS 抽出を試みる必要がなくなる。MFCS_Mining_Initialize()の 9～10 行目で Chemical Graph を要素サイズが昇順になるように並び替えることで、15 行目から始まる抽出処理において複雑な Chemical graph からの MCFS の抽出を回避させることで、計算コストを低減させる。

Support = 1 の場合に有効な探索打ち切り処理

Support = 1 の Subgraph の中で、MFCS となりうるのは Subgraph 抽出元の Chemical graph そのものだけである(第 2.2 節[定理 6])。よって、MFCS_Mining_Initialize()の 12 行目で MinSup=2 を設定し、MFCS_Mining()により MFCS を抽出する際、MFCS_Mining()の 16 行目で Support が 1 以下になった場合に探索を打ち切ることで、計算コストを低減させる。

この打ち切りは半数近くの Subgraph を計算対象から除外することが可能な非常に強力な打ち切りで、計算コストを大幅に低減させることが出来る。この処理で MFCS に含まれなかった Subgraph(Chemical graph そのもの)は MFCS_Mining_Initialize()の 28～34 行目で MFCS に追加する、1 つの化学構造から取得可能な Connected Subgraph は、数億や数十億になることもあるが、そこから取得できる MFCS の数は経験上ほとんどが 1～2 桁程度である。この為、MFCS_Mining_Initialize()の 28～34 行目の処理は MFCS_Mining()で MFCS を取得するのと比較すると計算コストは圧倒的に低い。

Parent subgraph と入力 Chemical graph の包含関係を利用した計算コストの低減

Result[ ][ ]には、探索で得られた Subgraph と Chemical graph との包含関係が格納されており、「探索で得られた Subgraph⊂Chemical graph」を満たす場合は 1、満たさない場合は 0 を記録している。配列の大きさは、「再帰呼び出しの最大深さ×Chemical graph の数」である。Result[0][ ]は null graph との包含関係が格納される為、全て 1 となる。Result[1][ ]は node 1 つからなる Subgraph との包含関係、それ以降は現在探索中の Subgraph に至るまでの各再帰呼び出しにおいて生成された Subgraph との包含関係が記録されている。MFCS_Mining()の 1～2 行目は、探索中の Subgraph の生成元となった Parent subgraph の包含関係をコピーしている。探索中の Subgraph は呼び出し元の Parent subgraph に隣接要素を追加することで得られるので、「Parent subgraph⊂探索中の Subgraph」である。よって、「Parent subgraph⊂TargetGraph」は「探索中の Subgraph ⊂TargetGraph」の必要条件となる。必要条件を満たさない Chemical graph を 6 行目で除外し、7 行目の比較処理を回避させることで、計算コストを低減させる。

探索中の Subgraph が過去に MFCS 抽出済みの Chemical graph に包含される場合

MFCS_Mining()の 7 行目で「探索中の Subgraph⊂TargetGraph」の関係が認められた場合、TargetGraph が MFCS_Mining_Initialize()の 15 行目の for ループ内で過去に MFCS を探索済みの Chemical graph であるかどうかを、MFCS_Mining()の 8 行目で調べている。TargetGraph が探索済みの Chemical graph であった場合、探索中の Subgraph が MFCS であるかどうかの判定は完了している為、SearchedSubgraph=1 を設定し、MFCS_Mining()の 58 あるいは 60 行目で関数を抜け、67～69 行目で探索中の Subgraph が MFCS に追加されないようにしている。

ここで、TargetGraph 以降の Chemical graph との比較を回避することで、計算コストを低減させる。この処理は、MFCS_Mining()の 13～15 行目で計算する Support の値に影響を及ぼすが、Support の値は、40 行目で Parent subgraph の Support と比較することで MFCS の対象であるかどうかの判定に使われる、ここで「Parent subgraph⊂探索中の Subgraph」である為、「探索中の Subgraph⊂過去に MFCS を探索済みの Chemical graph」の時、「Parent subgraph⊂過去に MFCS を探索済みの Chemical graph」が常に成立し、Parent Subgraph も出力対象外となるため問題は発生しない。

探索木上における、末端 Subgraph と包含関係にある Subgraph に着目した打ち切り

探索木の末端に現れる Subgraph は、これ以上追加可能な edge が存在しない Subgraph である為、MFCS_Mining()の 27～30 行目の判定処理のどれかで棄却され、31～54 行目の処理は実行されない。結果、58 行目に到達した時、ChildSubgraphNum の値は 20 行目で設定した 0 を保持する。ここで、「末端 Subgraph⊂過去に MFCS を探索済みの Chemical graph」であった場合、MFCS の対象から除外できるのは末端 Subgraph だけではなく、探索木上で末端 Subgraph のサブセットで構成される Subgraph に関しても、「末端 Subgraph のサブセットで構成される Subgraph⊂末端 Subgraph⊂過去に MFCS を探索済みの Chemical graph」が成立し、「末端 Subgraph のサブセットで構成される Subgraph⊂過去に MFCS を探索済みの Chemical graph」が成立するので MFCS の対象から除外することが可能である。Subgraph A から隣接 edge を追加する作業を繰り返すことで末端 Subgraph B に到達した場合、末端 Subgraph B を生成する過程で生成された全ての Subgraph が 28～54 行目の処理で 1 番初めに採用された Child subgraph であった場合(つまり ChildSubgraphNum=0 の場合)、Subgraph A 以降の探索で得られる全ての Subgraph は末端 Subgraph B のサブセットから得られる Subgraph となる。MFCS_Mining()の 58 行目で末端の Subgraph B、かつ、探索済みの Chemical graph に包含されていた Subgraph であるかどうかの判定を行い、該当した場合に「末端 Subgraph B⊂過去に MFCS を探索済みの Chemical graph」であったことを、Parent subgraph C に伝えることを目的として、BreakFlg=1 を設定した後、関数を抜ける。Parent subgraph C は末端 Subgraph B が設定した BreakFlg を BreakChk で受け取り、BreakChk=1 の場合、52 行目で別の edge を追

加することで生成可能なSubgraphを入力としたMFCS_Mining()の実行を行うことなく探索を打ち切る。さらに、末端Subgraph Bが、Parent subgraph Cに隣接edgeを追加することで得た1つ目のChild subgraphであった場合、Parent subgraph CのParent subgraph Dも MFCS対象外であることが確定するので、59行目で BreakFlgに1を設定した後、関数を抜ける。

制限時間による探索の打ち切り

以上の計算コスト低減処理をアルゴリズムに組み込んでも、MFCS 抽出アルゴリズムは根本的にNP困難な問題である為、Chemical graphが複雑なケースで計算が完了しない場合がある、MFCS_Mining_Initialize()の 11行目で一つの Chemical graphから MFCSを取得する為の制限時刻を設定し、MFCS_Mining()の 17 行目で制限時刻に到達したら探索を打ち切っている。

MFCS 抽出の例

以上に述べた手法を用いて、図 29の7個の Chemical graph g1～g7を例に MFCSの抽出を試みると、図 41の34個の MFCSを得ることが出来る。図中の番号は Subgraph IDを示しており、g1: Subgraph 1、g2: Subgraph 3、g3: Subgraph 16、g4: Subgraph 5、g5: Subgraph 34、g6: Subgraph 28、g7: Subgraph 25がそれぞれ元の Chemical graphと同形な Subgraphに対応している。抽出した34個の Subgraphに対して、この後第4.2節で説明するアルゴリズムを用いて Parent-Childの関係を取得し、代謝経路予測を行った結果が図 42である。ここで、予測した代謝経路は明らかに冗長である。例えば、図 42の緑色で示している Subgraphは2つの六員環から構成される Frameworkから edgeをいくつか取り除いた Subgraphである。第5.1.1項で検証を行うが、既知の代謝反応の98%において Framework間に包含関係が存在する。一度の代謝反応で環の生成と展開が同時に起こるような代謝反応は非常にまれなケースである。図 42の緑色で示している Subgraphはこの様な代謝反応を表現しており、代謝反応を表現する Subgraph としては不適切であり対象外としても問題はない。また、今回のケースでは、Chemical graph の構造が単純である為、例えば g1から MFCSを取得する際、探索木には60個の Unique Connected subgraphのみが出現し、一瞬で代謝経路予測結果を得ることが出来るが、Chemical graph が複雑な Frameworkを含むケースでは、図 28で説明したように膨大な数の Subgraphが存在する。その結果として、制限時間による打ち切りが大量に発生し、予測結果が曖昧なものとなる。

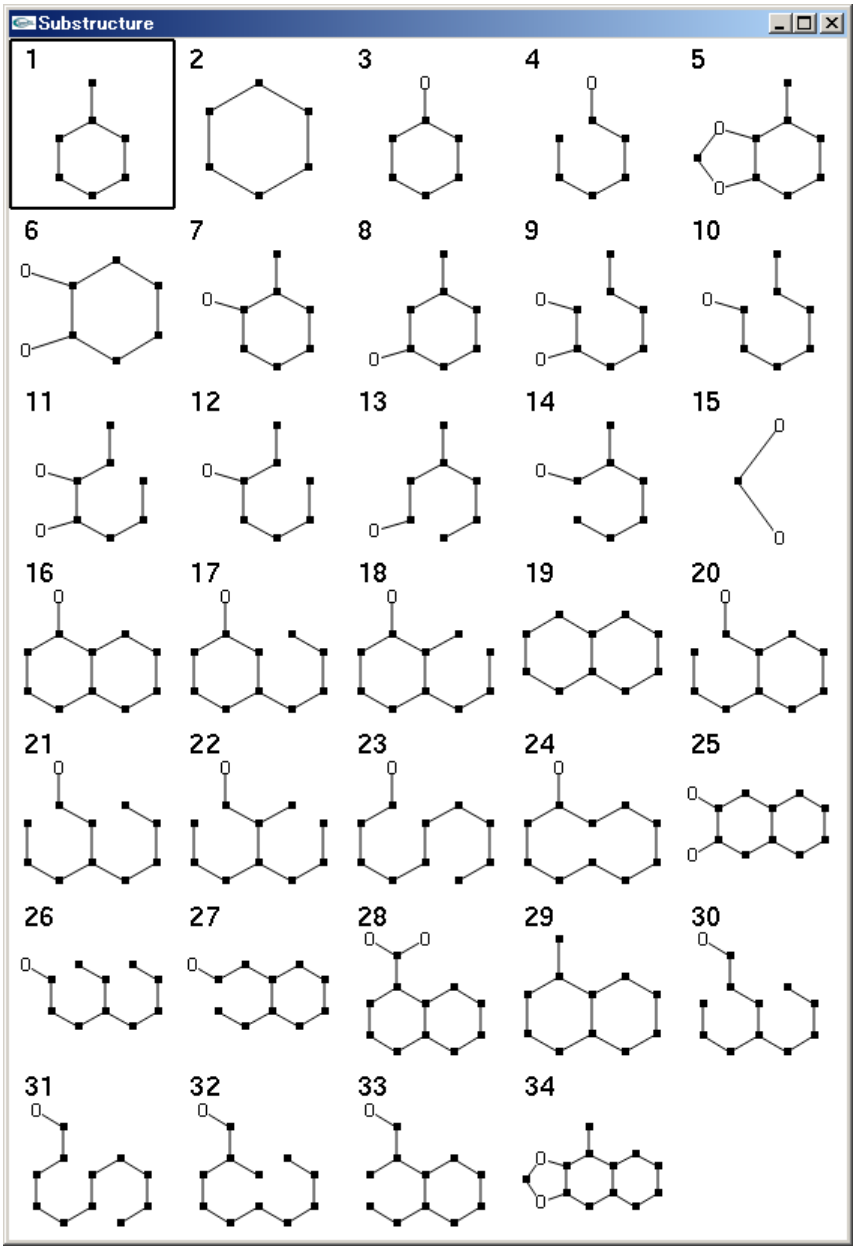


図 41 Maximum Frequent Connected subgraph 抽出結果

抽出した MFCS の内、入力とした 7 個の Chemical graph と同形なのはそれぞれ、g1: Subgraph 1、g2: Subgraph 3、g3: Subgraph 16、g4: Subgraph 5、g5: Subgraph 34、g6: Subgraph 28、g7: Subgraph 25 である。MFCS には入力とした Chemical graph と同形な Subgraph が必ず含まれる。よって、抽出した MFCS に対して代謝経路予測を行うことで Compound レベルの代謝経路予測が実現できる。

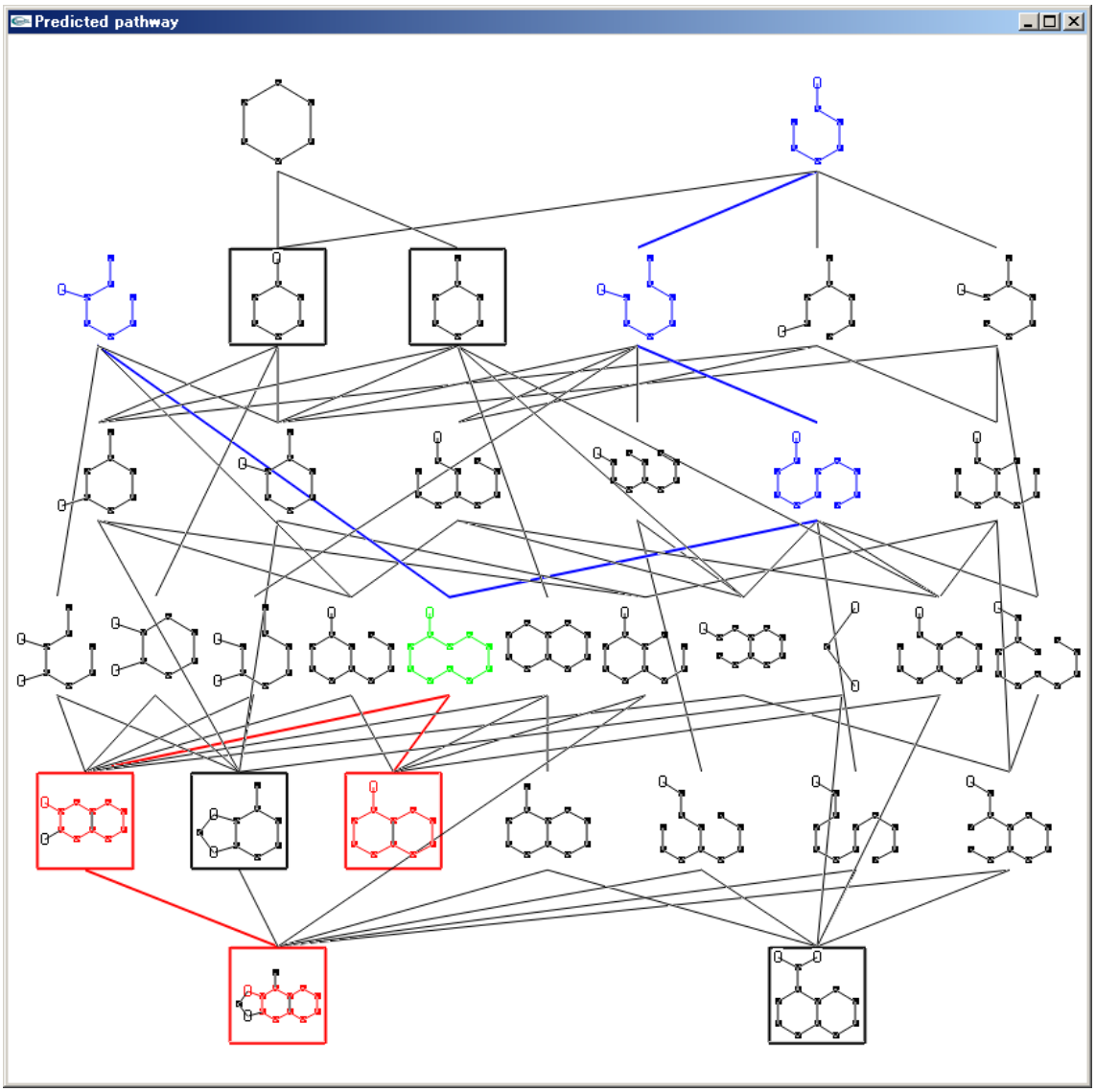


図 42 抽出した MFCS の包含関係に着目した代謝経路予測結果

図 41 の 34 個の MFCS に対して、第 4.2 節で説明を行う包含関係に着目した代謝経路予測を行った結果[8]。枠が付いた Subgraph は入力とした Chemical graph と同形の Subgraph である。ここで、予測した代謝経路は明らかに冗長である。例えば、図中緑色の Subgraph は 2 つの六員環から構成される Framework から edge をいくつか取り除いた Subgraph である。第 5.1.1 項で検証を行うが、既知の代謝反応の 98%では Framework 間に包含関係が認められる。Framework に含まれる edge を取り除くことで生成される Subgraph は代謝反応を表現する Subgraph としては冗長である。

[8] 本研究で開発したアルゴリズムはソフトウェア MetClassifier に実装されており、WEB 上で公開している。描画した代謝経路上の Subgraph を選択すると選択した Subgraph が緑色に、Ancestor subgraph が青色に、Descendant subgraph の中で選択した Subgraph と同形の部分が赤く表示される。図中四角の枠で囲っている Subgraph は、入力とした Chemical graph と同形のとれた Subgraph を意味している。

```
MFCS_Mining_Initialize(MFCS, G, TimeLimit)
/***** Parameter *****/
MFCS            /* Maximum Frequent Connected Subgraph */
G               /* A set of chemical graph */
TimeLimit       /* Time limit for extracting MFCS from one chemical graph */

0:{
/***** Count maximum element size *****/
1:MaxNodeNum ← Maximum # of node of chmical graph in G;
2:MaxEdgeNum ← Maximum # of edge of chmical graph in G;
3:MaxSearchTreeDepth ← MaxEdgeNum + 2;
/***** Memory allocate & Initialize *****/
4:IgnoreNode[ ] = allocate(MaxNodeNum);
5:IgnoreEdge[ ] = allocate(MaxEdgeNum);
6:Subgraph. node[ ] = allocate(MaxNodeNum);
7:Subgraph. edge[ ] = allocate(MaxEdgeNum);
8:Result[ ][ ] = allocate(MaxSearchTreeDepth, # of chmical graph in G);

/***** Sort G in ascending order for fast computing *****/
9:Sort G by # of node;
10:Sort G by # of edge;

/***** Initialize *****/
11:TimeoutClock ← clock( ) + TimeLimit;
12:MinSup ← 2;
13:MFCS ← ∅;
14:for( ii = 0 ; ii < # of chemical graph in G ; ii + + ) Result[0][ii] ← 1;
```

```
/***** MFCS mining *****/
15:for( ii = 0 ; ii < # of chemical graph in G – MinSup  +  1; ii + + ){
        /***** Set QueryGraph *****/
16:     QueryGraph ← G[ii];

        /***** Initialize IgnoreNode *****/
17:     for( NodeID = 0 ; NodeID < # of node in QueryGraph ; NodeID + + )
18:             IgnoreNode[NodeID] ← 0;

        /***** Start MFCS mining from QueryGraph *****/
19:     for( NodeID = 0 ; NodeID < # of node in QueryGraph ; NodeID + + ){
                /***** Set node as an initial subgraph *****/
20:             Subgraph ← QueryGraph. node[NodeID];
21:             IgnoreNode[NodeID] ← 1;
                /***** Initialize *****/
22:             BreakChk ← 0;
                /***** Start MFCS mining *****/
23:             MFCS_Mining(MFCS, G, QueryGraph, Subgraph, 1, Result,
                        IgnoreNode, IgnoreEdge, BreakChk, MinSup, TimeoutClock);
                /***** Update IgnoreNode for getting unique subgraph *****/
24:             IgnoreNode[NodeID] ← −1;
                /***** Terminate mining *****/
25:             if(BreakChk == 1)break;
26:     }
27:}

/***** Add Subgraph, which have all element of Chemical graph, to MFCS *****/
28:for( ii = 0 ; ii < # of chemical graph in G ; ii + + ){
29:     if(there are not exist isomorpic Subgraph with G[ii] in MFCS){
30:             Subgraph  ← All nodes of G[ii] ∪ All edges of G[ii];
31:             Subgraph. Support ← 1;
32:             MFCS ← MFCS ∪ Subgraph;
33:     }
34:}
35:}
```

```
MFCS_Mining(MFCS, G, QueryGraph, Subgraph, Depth, Result,
                              IgnoreNode, IgnoreEdge, BreakFlg, MinSup, TimeoutClock)
/***** Parameter *****/
MFCS            /* Maximum Frequent Connected Subgraph */
G               /* A set of chemical graph */
QueryGraph      /* Subgraph is extracted from this graph*/
Subgraph        /* Subgraph */
Depth           /* Depth of search tree*/
Result          /* Result of inclusive relation */
IgnoreNode      /* -1: Ignore node, >1: used node in Subgraph, 0: unused node */
IgnoreEdge      /*    0: Ignore edge or used edge in Subgraph, 0: unused edge */
BreakFlg        /* If BreakFlg become 1 then terminate mining */
MinSup          /* Minimum support */
TimeoutClock    /* Timeout clock for extracting MFCS from one chemical graph */

0:{
/***** Check inclusive relation *****/
1:for( ii = 0 ; ii < # of chemical graph in G ; ii + + )
2:      Result[Depth][ii] ← Result[Depth − 1][ii];
3:SearchedSubgraph ← 0;
4:for( ii = 0 ; ii < # of chemical graph in G ; ii + + ){
5:      TargetGraph ← G[ii];
6:      if(Result[Depth][ii] == 0) continue;
7:      if(Subgraph  ⊆ TargerGraph){
8:          if(TargetGraph is in front of QueryGraph in G){SearchedSubgraph ← 1; break; }
9:      }else{
10:         Result[Depth][ii] ← 0;
11:     }
12:}
/***** Count Support *****/
13:Support ← 0;
14:for( ii = 0 ; ii < # of chemical graph in G ; ii + + )
15:     if(Result[Depth][ii] == 1) Support + +;
/***** Terminate mining *****/
16:if(Support < Minsup) return Support;
17:if(clock( ) > TimeoutClock) return Support;
```

```
/***** Initialize mining parameter *****/
18:OutputFlg ← 1;
19:BreakChk ← 0;
20:ChildSubgraphNum ← 0;

/***** Continue mining *****/
21:for( EdgeID = 0 ; EdgeID < # of edge in QueryGraph ; EdgeID + + ){
        /***** Set Element and Element ID *****/
22:     Edge ← QueryGraph. edge[EdgeID];
23:     Node0 ← Edge. ConnectedNode0;
24:     Node1 ← Edge. ConnectedNode1;
25:     NodeID0 ←  Array number of NodeID0 in QueryGraph. node[ ];
26:     NodeID1 ←  Array number of NodeID1 in QueryGraph. node[ ];

        /***** Check connectivity for getting connected subgraph *****/
27:     if(IgnoreNode[NodeID0] == 0 && IgnoreNode[NodeID1] == 0)continue;
        /***** Check ignore element for getting unique subgraph *****/
28:     if(IgnoreEdge[EdgeID]! = 0) continue;
29:     if(IgnoreNode[NodeID0] == −1) continue;
30:     if(IgnoreNode[NodeID1] == −1) continue;

        /***** Update Subgraph & IgnoreNode & IgnoreEdge *****/
31:     Subgraph ← Subgraph ∪ Edge;
32:     IgnoreEdge[EdgeID] ← Depth + 1;
33:     if(IgnoreNode[NodeID0] == 0){
34:             Subgraph  ← Subgraph ∪ Node0;
35:             IgnoreNode[NodeID0] ← Depth + 1;
36:     }else if(IgnoreNode[NodeID1] == 0){
37:             Subgraph ← Subgraph ∪ Node1;
38:             IgnoreNode[NodeID1] ← Depth + 1;
39:     }
        /***** Recursive call *****/
40:     if(Support == MFCS_Mining(MFCS, G, QueryGraph, Subgraph, Depth + 1,
                Result, IgnoreNode, IgnoreEdge, BreakChk, MinSup, TimeoutClock){
41:             OutputFlg ← 0;
42:     }
```

```
        /***** Restore Subgraph & IgnoreNode *****/
43:     Subgraph ← Subgraph − Edge;
44:     if(IgnoreNode[NodeID0] == Depth + 1){
45:             Subgraph ← Subgraph − Node0;
46:             IgnoreNode[NodeID0] ← 0;
47:     }
48:     if(IgnoreNode[NodeID1] == Depth + 1){
49:             Subgraph ← Subgraph − Node1;
50:             IgnoreNode[NodeID1] ← 0;
51:     }
        /***** Terminate mining *****/
52:     if(BreakChk == 1)break;
53:     ChildSubgraphNum + +;
54:}
/***** Restore IgnoreEdge *****/
55:for( EdgeID = 0 ; EdgeID < # of edge in QueryGraph ; EdgeID + + ){
56:     if(IgnoreEdge[EdgeID] == Depth + 1) IgnoreEdge[EdgeID] ← 0;
57:}

/***** Local reduction *****/
58:if(SearchedSubgraph == 1&&ChildSubgraphNum == 0){BreakFlg ← 1; return Support; }
59:if(BreakChk == 1 && ChildSubgraphNum == 0){BreakFlg ← 1; return Support; }
60:if(SearchedSubgraph == 1) return Support;
61:if(OutputFlg == 0) return Support;
/***** Global reduction *****/
62:for( SubgraphID = 0 ; SubgraphID < # of Subgraph in MFCS ; SubgraphID + + ){
63:     CheckSubgraph ← MFCS[SubgraphID];
64:     if(Support == CheckSubgraph. Support && Subgraph ⊂ CheckSubgraph)
65:             return Support;
66:}

/***** Add Subgraph to MFCS *****/
67:Subgraph. Support ← Support;
68:MFCS ← MFCS ∪ Subgraph;
69:return Support;
70:}
```

第4.1.3項　Framework-based Maximum Frequent Connected Subgraph

第 4.1.2 項で実装した計算コスト削減処理は、制限時間を設定する処理を除いて、処理の導入前後で取得できる MFCS が変化しない処理であり、この様な処理結果に影響を及ぼさない処理はアルゴリズムの完全性を保持した処理であると呼ばれ、一見すると全ての計算が完了することが出来れば、精度の高い代謝経路予測が実現できるように感じてしまう。しかし、図 42 で説明した予測結果を見てわかる通り、全ての Connected subgraph を対象とした MFCS の中には、代謝反応の表現として冗長な Subgraph が多く含まれる。また、複雑な Chemical graph を含むケースでは、制限時間による打ち切りにより、本来 MFCS に含むべき Connected subgraph のいくつかが、MFCS から除外されてしまう為、代謝経路予測結果が曖昧になる。ヒューリスティックを用いない Maximum Common Subgraph 抽出と Tanimoto 係数による代謝経路予測のケースと同様に、膨大な計算コストをかけて仮に完全解を得ることが出来たとしても、予測結果が曖昧であったり冗長であったりしては、計算コストをかける意味がない。そこで、本研究では既知の代謝経路において基質と生成物の Framewrok 間に包含関係が認められる点をヒューリスティックとして導入し、Framework-based Connected subgraph のみを対象とした MFCS（Framework-based MFCS)を、代謝経路を表現する為の中間構造として採用した。図 28 で説明したように、Framework が複雑な Chemical graph からは膨大な数の Connected subgraph が取得可能であり、この様な Chemical graph が含まれるデータセットに対して MFCS 抽出を試みた場合、膨大な計算コストを必要とするが、Framework-based Connected subgraph のみに着目することで、対象となる Subgraph の数が激減するので、大幅な計算コストの削減が可能となる。また、この様なヒューリスティックの導入は、MFCS を取得するアルゴリズムと言う観点からみれば、完全解が得られない手法ではあるが、ヒューリスティックの導入により MFCS から除外される Subgraph は、代謝反応を表現する Subgraph としては不適切な Subgraph である為、より望ましい代謝経路予測結果を得ることが可能となる。

Framework-based MFCS を取得する際には、Chemical graph が Framework を持たない鎖状構造のケースと Framework を持つケースで初期 Subgraph の扱いが異なる。Framework が存在しない鎖状構造の Chemical graph に関しては、第 4.1.2 項の手法と同様に、Chemical graph に含まれる各 node を初期 Subgraph とする。Framework が存在する Chemical graph に関しては、第 4.1.1 項の手法で得ることが出来る Framework を初期 Subgraph とする。初期 Subgraph を用いて、MFCS_Mining()を呼び出すことで Framework-based MFCS の取得を行う。

以上に述べた手法を用いて、図 29 の 7 個の Chemical graph g1～g7 を基に MFCS の抽出を試みると、図 43 の 10 個の Framework-based MFCS を得ることが出来る。このケースでは、g1: Subgraph 1、g2: Subgraph 2、g3: Subgraph 8、g4: Subgraph 4、g5: Subgraph 10、g6: Subgraph 9、g7: Subgraph 6 がそれぞれ元の Chemical graph と同形な Subgraph に対応している。抽出した 10 個の Framework-based MFCS に対して、第 4.2 節で説明す

るアルゴリズムを用いて Parent-Child の関係を取得し、代謝経路予測を行った結果が図 44 である。図 42 の全ての Connected subgraph を対象とした MFCS を抽出したケースと比較して、代謝経路を表現する中間構造として不適切な MFCS が出現せず、望ましい代謝経路が予測できている。

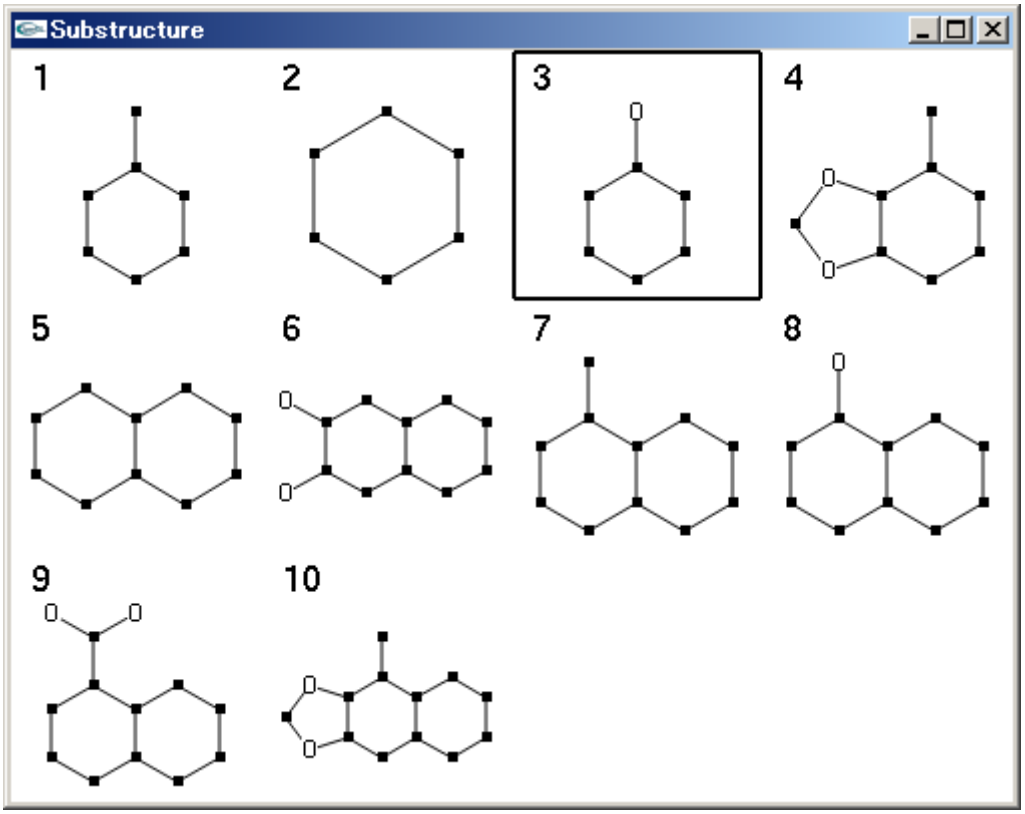


図 43 Framework-based MFCS 抽出結果

図 29 の 7 個の Chemical graph から 10 個の Framework-based MFCS を取得した。抽出した MFCS の内、入力とした 7 個の Chemical graph と同形なのはそれぞれ、g1: Subgraph 1、g2: Subgraph 2、g3: Subgraph 8、g4: Subgraph 4、g5: Subgraph 10、g6: Subgraph 9、g7: Subgraph 6 である。MFCS と同様に Framework-based MFCS にも入力とした Chemical graph と同形な Subgraph が必ず含まれる。よって、抽出した Framework-based MFCS に対して代謝経路予測を行うことで Compound レベルの代謝経路予測が実現できる。

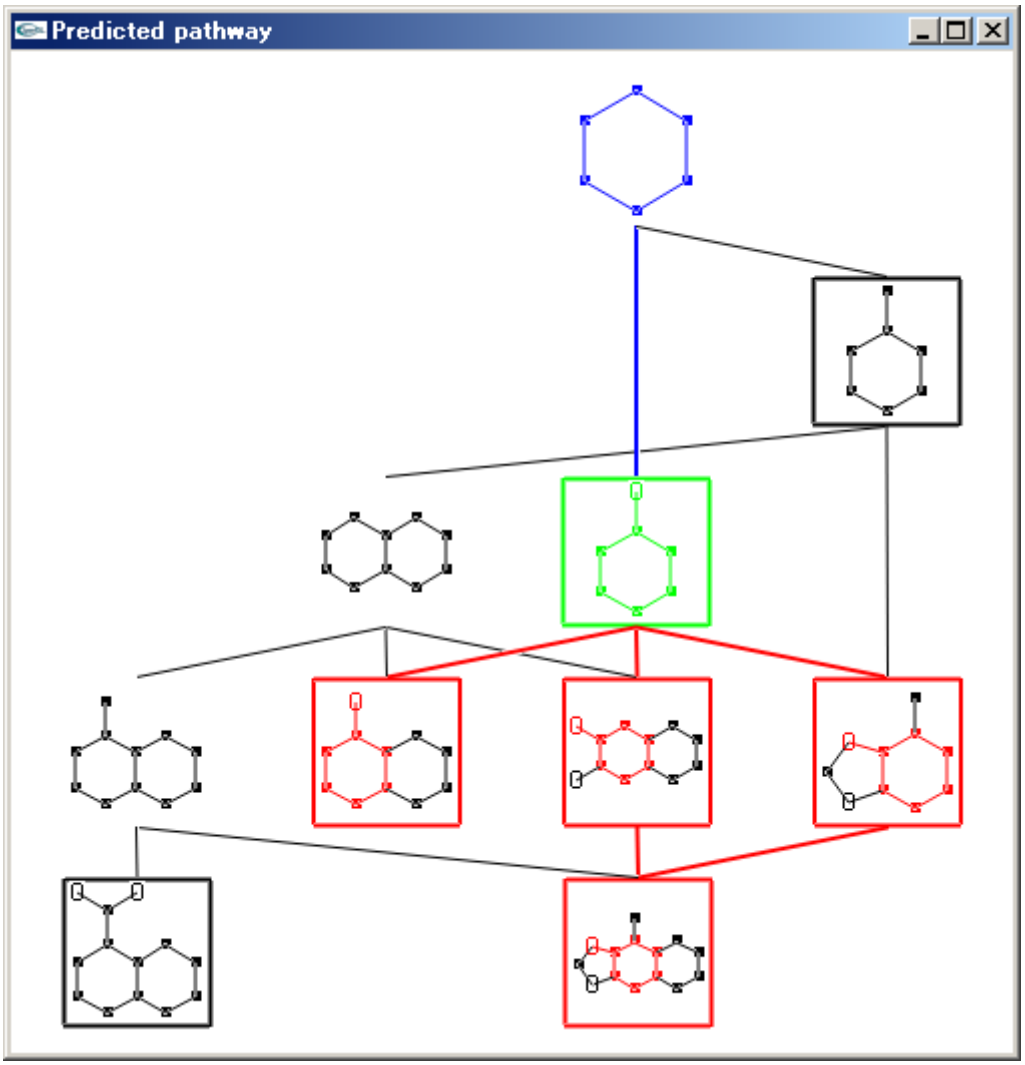


図 44 抽出した Framework-based MFCS の包含関係に着目した代謝経路予測結果

第4.2節　　Pathway prediction focusing on inclusive relation of subgraphs

包含関係に着目した代謝経路予測は以下の手順で行う。

1. M 個のユニークな Subgraph のセット USG(M)を入力とする。
2. USG(M)自身を Fragment library とし、各 USG(M)の Fingerprint を計算する。
3. 各 USG(M)の Parent Subgraph を計算する。
4. Parent-Child の関係を Subgraph レベルの代謝経路として出力する。

図 45 に、第 4.1.1 項の図 30 で例題として取得した 4 個の Framework を入力として包含関係に着目した代謝経路予測を行った結果を示す。

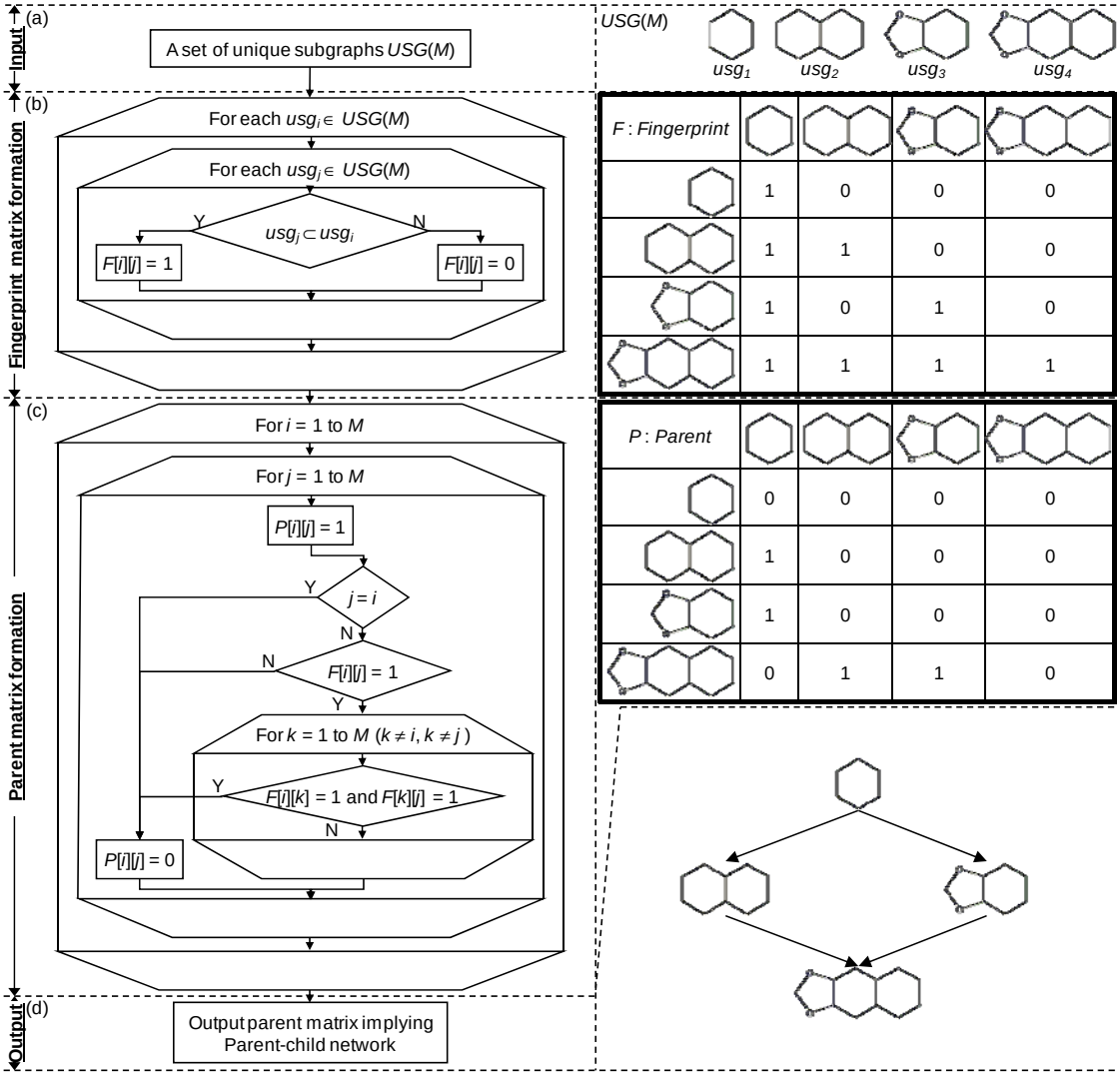


図 45 包含関係に着目した代謝経路予測

図 30 で取得した Framework のセットを USG(M)として用いている。

ここで、図 45 処理(b)および処理(c)は計算速度を優先させた手法である。処理(c)において Parent subgraph を見つける際に、同じ Subgraph 間の部分構造検索が複数回実行される。処理(b)において全 Subgraph 間の包含関係を事前に全て計算しておき、処理(c)では結果を用いることで計算コストの低減を行っている。しかし、このアプローチは対象とする Fingerprint を格納する為に、(Subgraph の数)^2 のデータ領域が必要となる。第 5 章において提案手法を用いた代謝経路予測を行っているが、各項において代謝経路予測時に入力となる Subgraph の数は以下の様になる。

第 5.1.1 項　　536 Framework
第 5.1.2 項　　21 Framework-based MFCS
第 5.1.3 項　35,653 Framework-based MFCS
第 5.2.1 項　　181 Framework-based MFCS
第 5.2.2 項 690,695 Framework-based MFCS

第 5.2.1 項までは、問題なく動作するが、第 5.2.2 項の 690,695 Framework-based MFCS を処理する為には、690,695×690,695＝477,059,583,025 のデータ領域が必要になる。これは、Fingerprint 1bit を表現するのに、char 型(1byte)を用いたとしても、約 480GB のメモリが必要になる計算であり、動作させることができない。

解決策として、最終的な実装では図 45 処理(b)および処理(c)の代わりに次ページで説明するメモリ使用量を優先させた手法を用いた。どちらの処理も目指しているのは Parent-Child の関係を取得することであり、同じ出力結果を得ることが出来るが、概念を説明する際に、前者のモデルは比較的シンプルであり理解しやすいので本論文に記述を残した。メモリ使用量を優先させた手法を用いた場合、KNApSAcK の全データ解析に対して要求されるメモリは、作業用の一時領域としての Descendant(M)および Child(M)に 4×690,695×2 =5,525,560　5.5MB（4 はポインタのデータサイズ）、各 Subgraph の Parent Subgraph および Child Subgraph を格納するデータ領域のみである。Parent Subgraph および Child Subgraph に関しては予測した代謝経路上で直接繋がっている Subgraph のみが対象データとなる為、データ量的にはわずかである。システム全体としては、700MB 程度のメモリで動作可能で、690,695 Subgraph 間の包含関係を比較し代謝経路を予測するのに Intel core2 Duo processor T7600 2.33GHz, 1GB RAM の計算環境を用いるとおよそ 2 週間で計算が完了する。第 5.2.2 項よりも解析対象データ量の少ない第 5.1.3 項では、35,653 Framework-based MFCS に対して同様の処理を行った場合、代謝経路予測にかかる時間は 30 分程度であった。

メモリ使用量を優先させた包含関係に着目した代謝経路予測

1. M 個のユニークな Subgraph のセット USG(M)を入力とする。
2. USG(M)を edge の数をキーに昇順でソートする（図 46)。
3. サイズ M の箱を 2 つ（Descendant(M)および Child(M)）を用意する（図 47)。
4. USG(M)の各要素 usg_i に(i = 1 to M)対して、処理 5～11 を実行する。
   5. usg_i をクエリ構造とし、usg_j(j = i+1 to M)に対して、部分構造検索を実行し、Descendant subgraph を取得する（L 個取得できたとする)。取得した L 個の Descendant subgraph を Descendant()にセットする（図 48)。
   6. 全ての $Descendant_k$(k = 1 to L)に child flg = 1 をセットする（図 49)。
   7. Descendant()の各要素 $Descendant_k$(k = 1 to L)に対して、処理 7～9 を実行する（図 50)。
      8. child flg = 0 なら次の要素の処理に移る。
      9. child flg = 1 ならば、usg_i と $Descendant_k$ の間には中間構造が存在しないことが確定するので、$Descendant_k$ を Child()に追加する。
      10. $Descendant_k$ をクエリ構造とし $Descendant_l$(l = k+1 to L)に対して部分構造検索を実行し、包含関係が認められた部分構造に child flg = 0 をセットする。
   11. 得られた Child()を usg_i の Child Subgraph として保存する。
12. Parent-Child の関係を代謝経路として出力する。

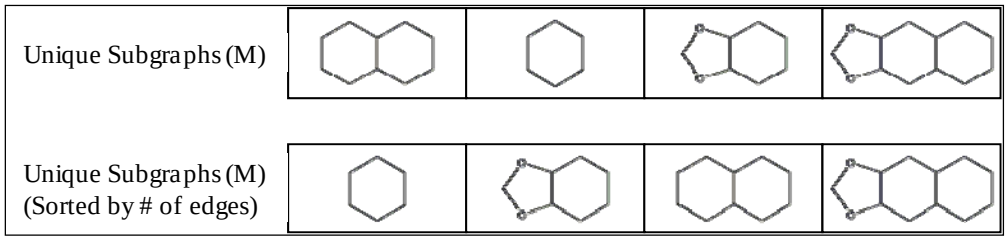


図 46 入力された Unique Subgraphs をエッジ数でソート

図 47 作業用領域の確保

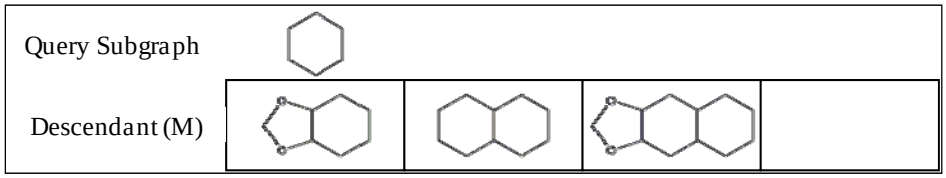


図 48 Descendant Subgraph の取得

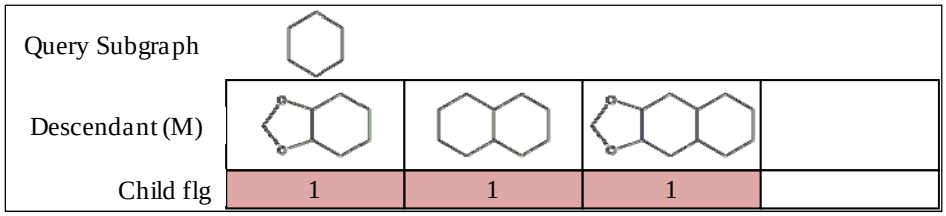


図 49 取得した Descendant Subgraph を Child Subgraph の候補とする。

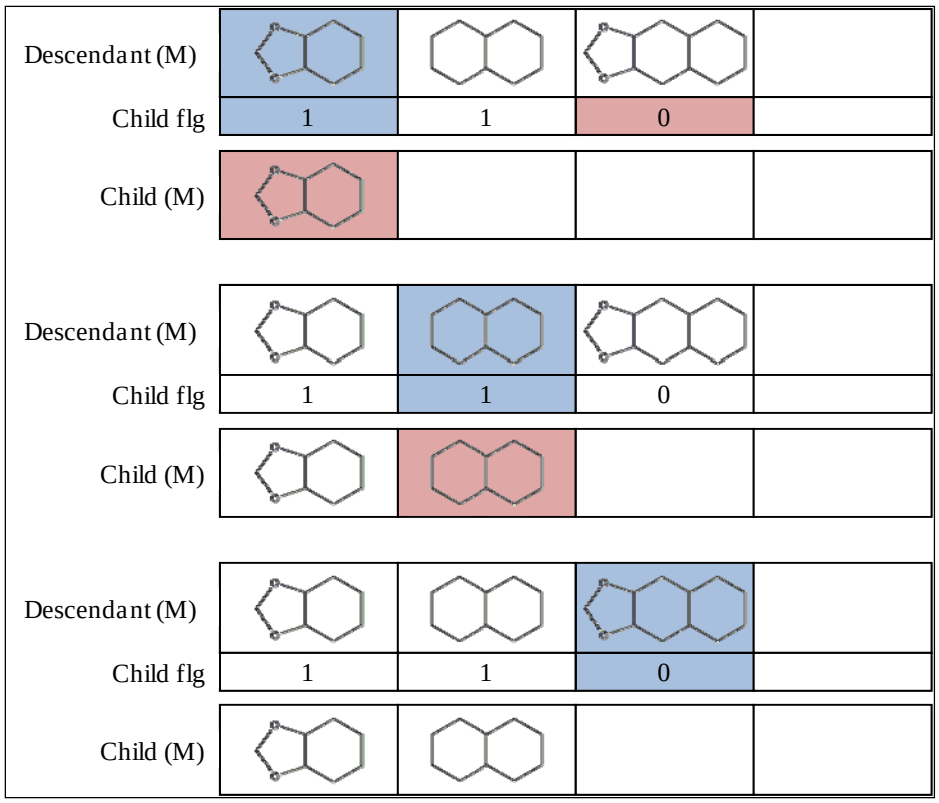


図 50 Descendant Subgraph から Child Subgraph を取得する。

第5章　結果と考察

本研究では、代謝経路予測を行う過程の MFCS 抽出アルゴリズムにおいて、既知の代謝経路上で隣り合う代謝物の Framework 間に包含関係が認められる特徴をヒューリスティックとして導入し、Framework を含む Connected subgraph のみを MFCS 抽出対象とすることで、計算コストの低減及び予測精度の向上を実現した。本章では、まず第 5.1 節において、Framework に着目したヒューリスティックの導入に妥当性があるかどうかを既知の代謝経路情報である KEGG の RPAIR を用いて検証した後、提案手法の予測精度を評価する。その後、第 5.2 節において、KNApSAcK を用い、生物種固有の代謝経路予測および大量のデータに対して提案手法が適用可能であるかどうかの検証を行う。なお、本章における計算速度の評価には Intel core2 Duo processor T7600 2.33GHz, 1GB RAM の計算環境を利用した。

第5.1節　　代謝経路が既知の代謝物を用いた提案手法の評価

本節では、KEGG の RPAIR に含まれる、代謝経路が既知の代謝物情報を用いて、提案手法の評価を行う。代謝反応を観察すると、反応前後において変化しない部分、付加または脱離によって変化する部分、前記の二つの境界領域にあって反応の中心的舞台と考えられる部分の 3 種類の情報を抽出できる。ここで、基質と生成物の二項関係がどの変換パターンに該当するのかをデータベース化したものが RPAIR である(Hattori et al., 2005)。第 5.1.1 項では、代謝経路が既知の基質と生成物の Framework 間においてどの程度の包含関係が存在するのかを検証する。第 5.1.2 項では、KEGG の Pathway データベースに含まれる Flavonoid 代謝に関連する一部の代謝物を用い、提案手法により代謝経路を予測した場合に、既知の代謝経路がどこまで正しく予測できるか比較を行う。第 5.1.3 項では RPAIR の全情報を用いて予測精度の検証を行う。

第5.1.1項　　Framework 包含関係の検証

本項では、代謝経路が既知の基質と生成物の Framework 間において、どの程度の包含関係が存在するのかを検証することにより、提案手法の中で Framework の包含関係をヒューリスティックとして導入していることの妥当性を証明する。テストデータには KEGG の RPAIR において main として定義されている 9,842 の代謝物ペアの内、代謝物ペア両方に関して構造データが存在しており、-R の様な省略記号が用いられていない 4,976 個の代謝物から構成される 6,131 代謝物ペア間の Framework に包含関係が認められるかどうか検証を行った。

4,976 個の代謝物から第 4.1.1 項の手法によって Framework を抽出したところ、536 個のユニークな Framework を抽出することが出来た(付録 A)。さらに、得られた 536 個のユニークな Framework の包含関係に着目して代謝経路予測を行ったところ、1,094 個の

Framework レベルの代謝経路(Parent-Child の関係が得られた Framework ペア)を予測することができた。Framework の抽出および Framework レベルの代謝経路予測はどちらも 1 秒未満で計算は完了した。RPAIR で定義されている代謝物ペアを、予測した Framework レベルの代謝経路に照らし合わせると、同一 Framework：66.9%(4,102/6,131)、Parent-Child:25.2%(1,544/6,131)、Ancestor-Descendant:6.1%(373/6,131) (付録 B)、包含関係が存在しない:1.8%(112/6,131) (付録 C)となった。つまり、全 6,131RPAIR の内、98%に相当する 6,019 RPAIR の Framework 間には包含関係が存在することが確認できた。このことから、Framework の包含関係に着目した代謝経路予測の妥当性が証明できた。包含関係の得られなかった 112 RPAIR の内訳は以下の通りである。

- 環拡大反応 計 59 件（図 51）
- 新しい環の構成 or 付加と展開 or 脱離が同時に起こっているケース 計 15 件（図 52）
- Framework を構成している原子が他の原子に変化するケース 計 10 件（図 53）
- 基質と生成物間の保存箇所が非連結なケース 計 8 件（図 54）
- Linker が伸長するケース 計 4 件（図 55）
- その他 計 14 件

基質と生成物間の保存箇所が非連結なケースに関しては、フラボノイドからイソフラボノイドに変化する反応が 7 件で、Framework ID105-107 のペアが 1 件である。Ring の付き方は変わるが代謝反応が認められないようなペアは無数に存在する為、構造情報のみから、これらを発見するようなアルゴリズムは False-Positive の増大に繋がる。

その他のケースに関しては、全ての Connected subgraph を対象として MFCS の抽出を行えば予測対象に含まれるが、このアプローチは(Framework-based ではない)MFCS を用いた代謝経路予測(図 42)で示したように膨大な計算コストが要求される。また、仮に全ての計算が完了出来たとしても代謝経路を表現するのにふさわしくない Subgraph が増加する為、False-Positive の増大にも繋がる。

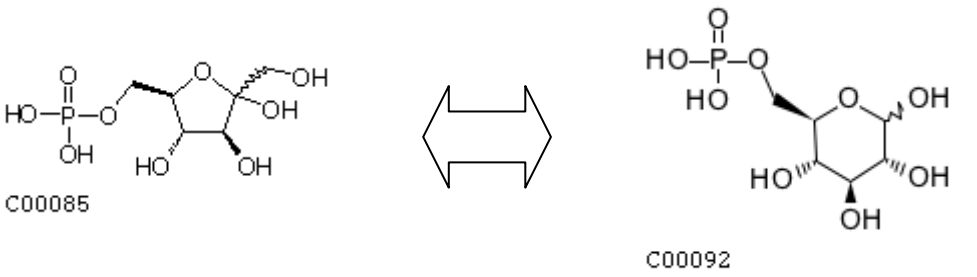


図 51 環拡大反応

RP01093(Framework ID 6-15) 同様のケースは 59 件

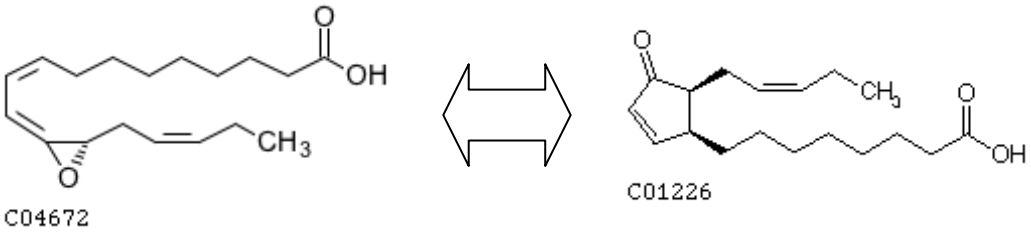


図 52 新しい環の構成 or 付加と展開 or 脱離が同時に起こっているケース

RP03016(Framework ID 2-11) 同様のケースは計 15 件

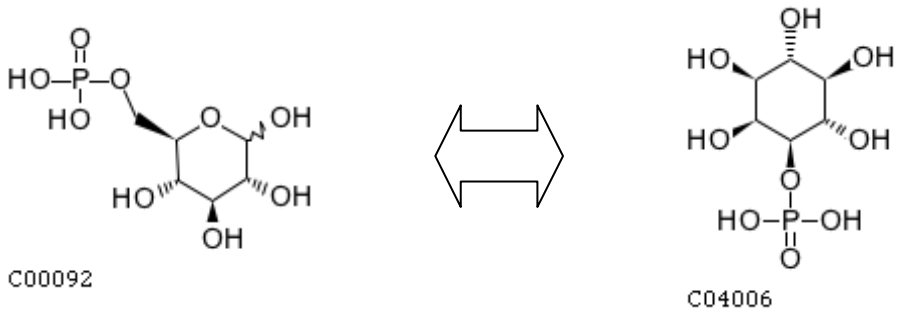


図 53 Framework を構成している原子が他の原子に変化するケース

RP10933(Framework ID 15-16) 同様のケースは計 10 件

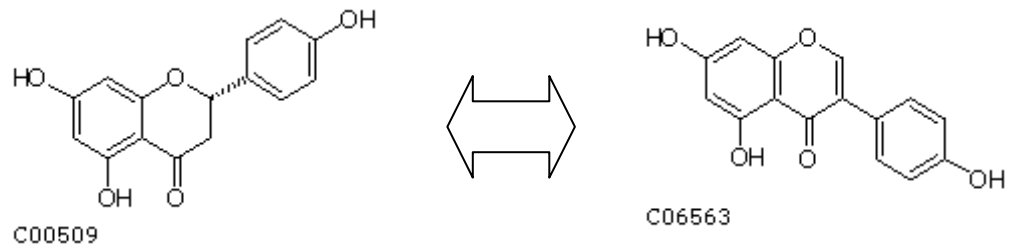


図 54 基質と生成物間の保存箇所が非連結なケース
RP09033(Framework ID 183-184) 同様のケースは計 8 件

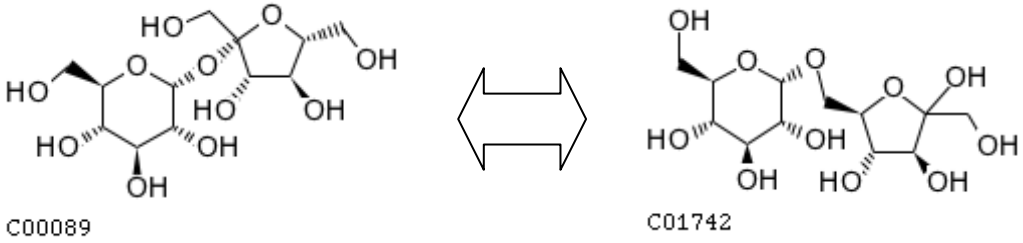


図 55 Linker が伸長するケース
RP01113(Framework ID 88-111) 同様のケースは計 4 件

第5.1.2項　　少数の類似化合物に対する予測結果を確認

既知の代謝経路として KEGG Pathway データベースから map00942 の一部(図 56)を用い、Framework レベルの代謝経路予測および Compound レベルの代謝経路予測を行い、提案手法の予測精度について評価を行う。

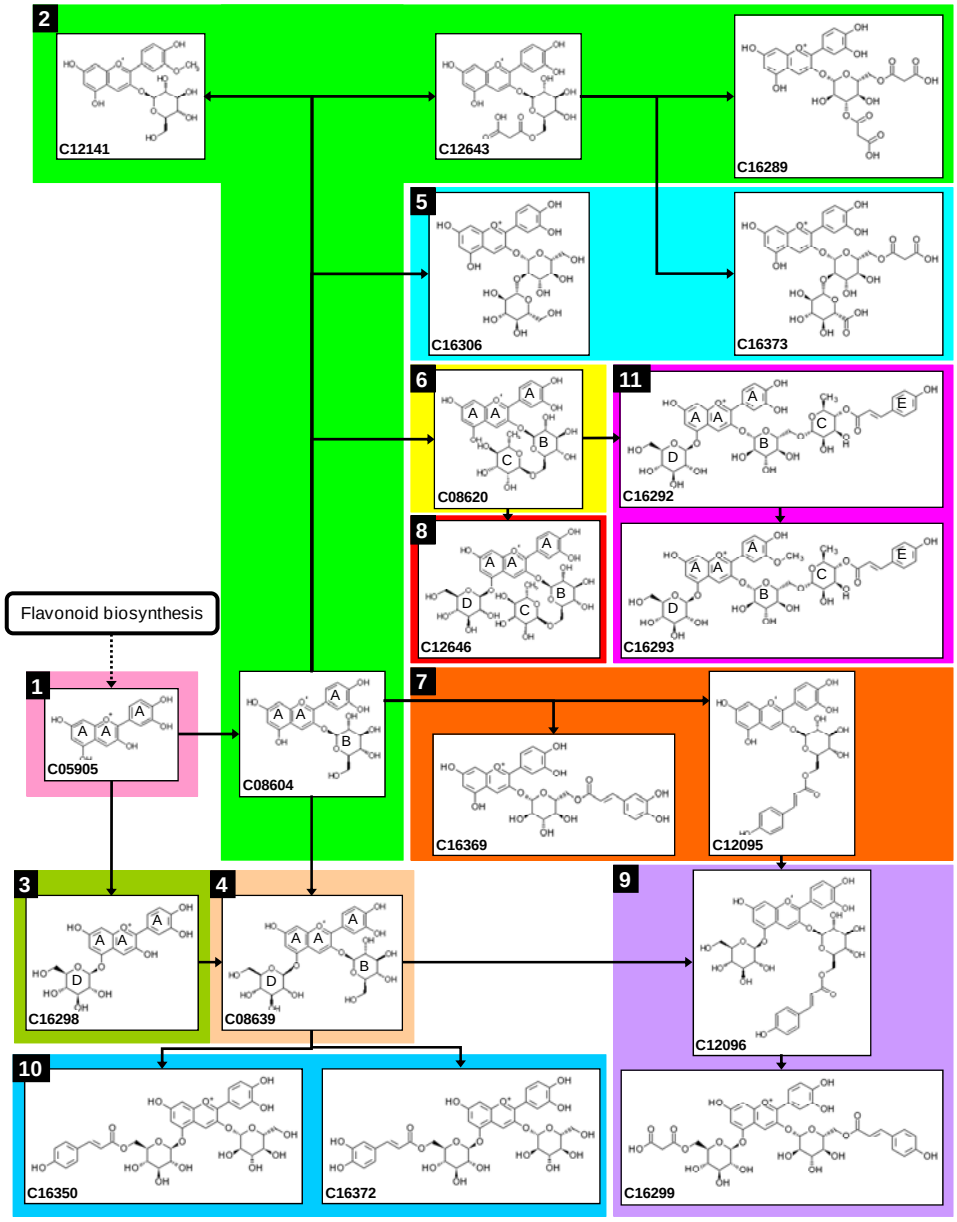


図 56 KEGG Pathway map00942 の一部

19 の化合物と 20 の代謝反応が含まれている。同じ Framework をもつ代謝物は背景色でクラスタリングして表示した。各クラスタの左肩の数字は Framework ID を表している。

Framework レベルの代謝経路予測

図 56 に含まれる 19 代謝物から 11 のユニークな Framework が抽出できた(図 57)。さらに、包含関係に着目した代謝経路予測を行ったところ、図 58 の Framework レベルの代謝経路予測結果を得た。

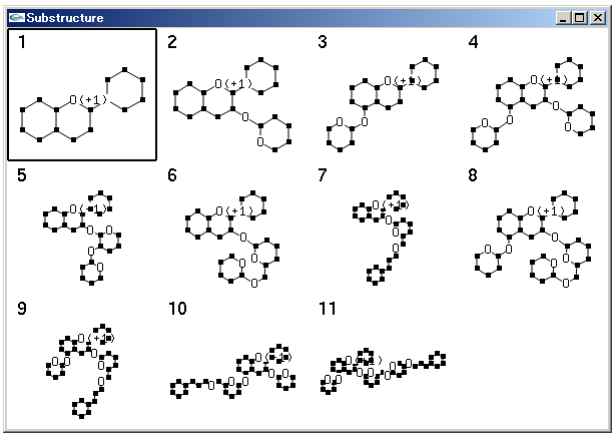


図 57 抽出した 11 のユニークな Framework

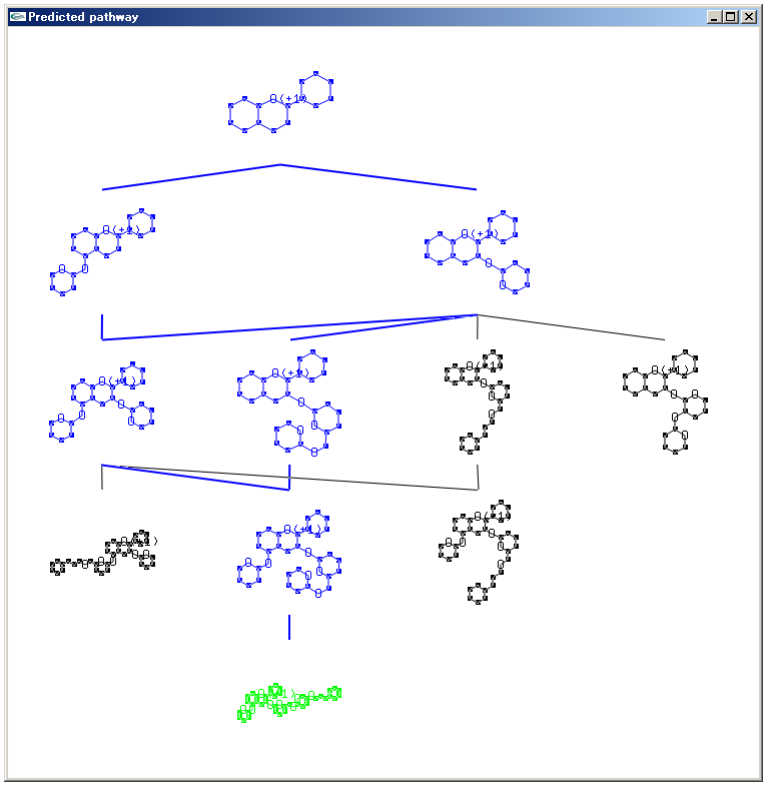


図 58 Framework レベルの代謝経路予測結果

既知の代謝経路から得られる Framework レベルの代謝経路と予測結果の比較を行う(図59)。(a)は既知の代謝経路を Framework レベルの代謝経路で表したものである。(b)は提案手法を用いて予測した Framework レベルの代謝経路を表している。(b)の赤の破線は予測に含まれなかった経路を示し、青線は新しく予測した経路を示す。Framework レベルの代謝経路に置き換えた場合、既知の経路には 12 の経路が存在する。提案手法で得られた結果は、11 の経路を正しく予測し、1 つの経路が予測できず、2 つの新規経路を発見した。

予測が出来なかった原因は、提案したアルゴリズムが FID(Framework ID)8 は FID 6 と FID 11 の中間構造であると認識した為である。その結果 FID 6→FID 11 という経路が棄却され、FID 6→FID 8→FID 11 の経路が採用されている。これらの原因を説明する為、図 56 に表示されている一部の代謝物の環に A～E の印をつけた。ここで、FID 6、8、11 に該当する化合物の環に着目してみると、FID 6 は ABC 環を持ち、FID 8 は ABCD 環を持ち、FID11 は ABCDE 環を持っている。つまり FID 8 は明らかに FID6 と FID11 の中間構造であり、予測結果は妥当であると言える。実際の代謝反応には、複雑な反応を一度に実現する酵素も存在するので、予測結果が正しいと断言することは出来ないが、化学構造の類似性から判断する予測としては最適な予測をしていると言える。新規に予測した経路は、FID 4 と FID 8 の間に中間構造となる構造が存在しなかった為 FID 4→FID 8 も新規の経路として予測した。こちらに関しても、FID 4 は ABD 環を持ち、FID 8 は ABCD 環を持っていることから、妥当な予測であると言える。

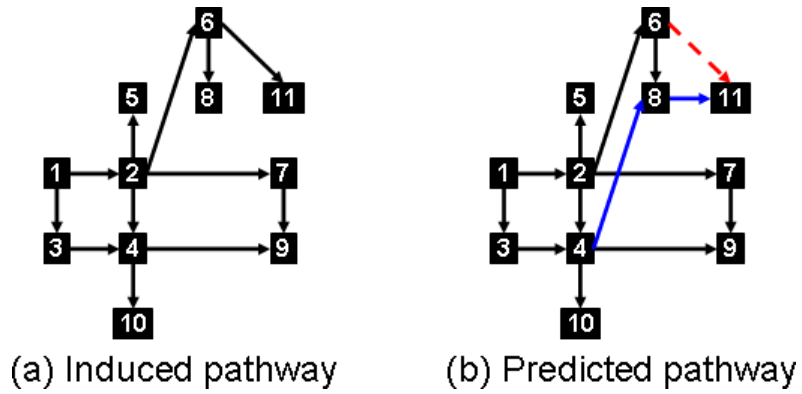


図 59 Framework レベルにおける代謝経路予測結果の評価

(a)は元の代謝経路を Framework レベルの代謝経路で表したものである。(b)は提案手法を用いて予測した Framework レベルの代謝経路を示している。(b)の赤の破線は予測に含まれなかった経路を示し、青の破線は新しく予測した経路を示す。提案したアルゴリズムは、FID(Framework ID) 8 は FID 6 と FID 11 の中間構造であると認識した。その結果 FID 6→FID 11 という経路よりも FID 6→FID 8→FID 11 の経路が尤もらしいと予測している。また、FID 4 と FID 8 の間に中間構造となる構造が存在しなかった為 FID 4→FID 8 も新規の経路として予測した。

Compound レベルの代謝経路予測

次に、Framework-based MFCS を用いた Compound レベルの代謝経路予測を行った。図 56 に含まれる 19 代謝物から 21 のユニークな Framework-based MFCS が抽出できた(図 60)。さらに、包含関係に着目した代謝経路予測を行うことで図 61 の Framework-based MFCS レベルの代謝経路予測結果を得た。計算時間は 1 秒程度であった。

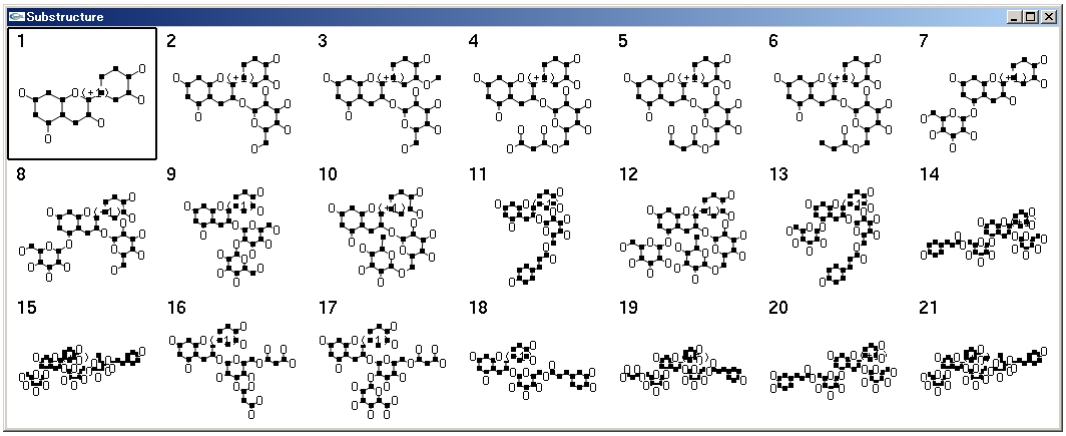


図 60 抽出した 21 のユニークな Framework-based MFCS

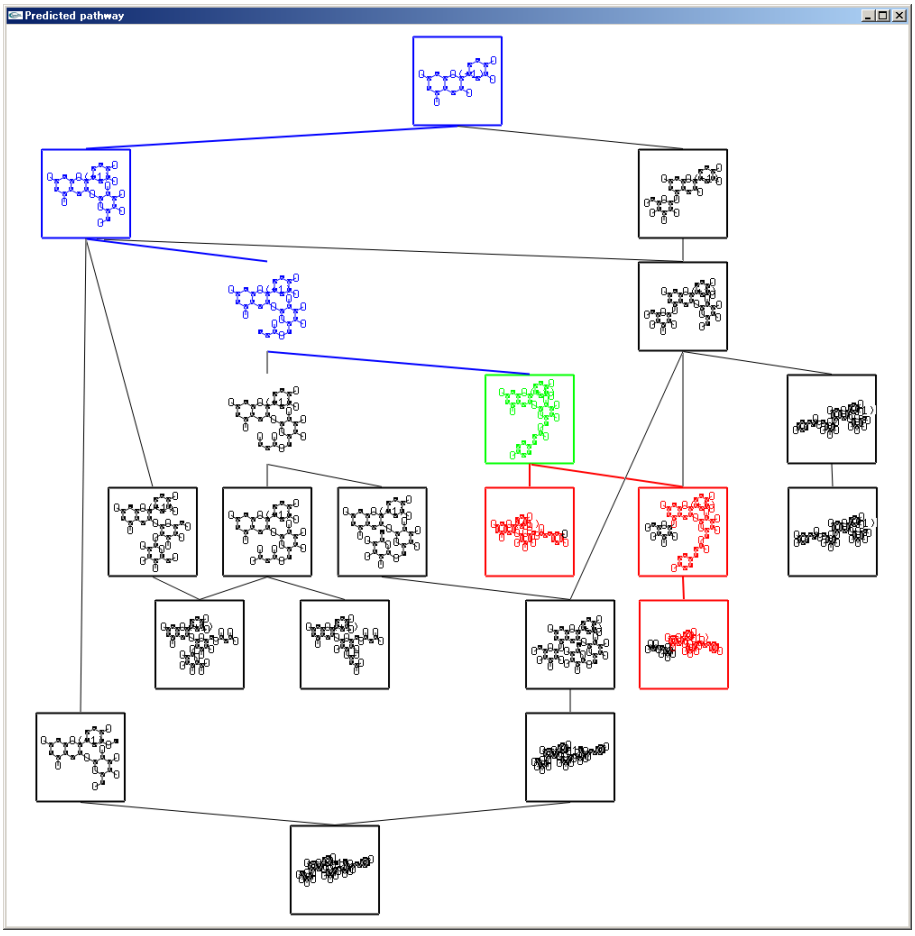


図 61 Framework-based MFCS レベルの代謝経路予測結果

既知の代謝経路と予測した代謝経路の比較

既知の代謝経路(図 56)と予測した代謝経路(図 61)を比較する為、両方の代謝経路の和を考えた場合に、全ての代謝経路が交差しないように図 56 の代謝物の配置を変更したのが図 62 の代謝経路である。図 61 で枠の付いている Framework-based MFCS は「入力とした Chemical graph と同形のとれた Framework-based MFCS」を意味している。図 63 は同形のとれた Framework-based MFCS を、対応する Chemical graph に置き換えた後、図 56 の代謝物と同じ配置で予測代謝経路を表示した結果である。青い太線が新規に予測した代謝経路、赤い破線が予測できなかった代謝経路、Subgraph 5 と Subgraph 6 は抽出した Framework-based MFCS の内、入力とした Chemical graph と同形は取れなかったが複数の Chemical graph が共通に持つ中間 Subgraph を表現している。ここで、予測結果はいくつかのパターンに分類できる。

(1) 既知の代謝経路上で隣り合う代謝物 A と代謝物 B の中間構造にあたる代謝物 C が発見されたケース。

予測した代謝経路	棄却された代謝経路
(C08639→)C16350→C16372	C08639→C16372
(C08620→)C12646→C16292	C08620→C16292
(C08604→Subgraph 6→)C12095→C16369	C08604→C16369

(2) 既知の代謝経路上で隣り合う代謝物 A と代謝物 B の中間 Subgraph C が発見されたケース。

予測した代謝経路	棄却された代謝経路
C08604→Subgraph 6→C12095	C08604→C12095
C08604→Subgraph 6→Subgraph 5→C12643	C08604→C12643
C08604→Subgraph 6→Subgraph 5→C08620	C08604→C08620

(3) 新規代謝経路を予測したケース。

予測した代謝経路	棄却された代謝経路
C08639→C12646	なし
C16306→C16373	なし
C12141→C16293	なし

(1)のケースは構造情報のみから予測する代謝経路としては最適解である。予測した代謝経路を棄却し、既知の代謝経路を正解と判定する為には、予測した代謝物ペアよりも化学構造が類似していない別の代謝物ペアを正解と判断する為の指標が必要であり、生化学や有機化学の知見を導入する必要がある。また、これらの知見を導入して予測した代謝経路に修正をかけることを考えた場合、予測結果を用いない場合は、19 代謝物の全組合せ 171 ペアを対象として、どの代謝物ペアが代謝経路上で隣り合っている代謝物であるのかを判定する必要が出てくるのに対し、提案手法により化学構造情報に基づいた代謝経路予測を行った後に、同様の処理を行う場合は予測代謝経路上で近い位置に存在する代謝物に目を向ければよくなる為、作業効率を大幅に改善することができる。

(2)のケースは、Subgraph を間に挟むことを許して直接到達できる代謝物のペアを予測代謝経路として出力することで、棄却された代謝経路を解答に含むことが出来る。また、抽出した中間 Subgraph に関しても、入力とした Chemical graph のセットには含まれないが、実在する代謝物と同形であるケースが存在する。生体から代謝物を抽出する際、処理の過程において酵素反応が進行し検出時には全て生成物に変化してしまい検出されないケースや文献で報告されていないが実在する代謝物であるケースがこれに相当する。

(3)のケースは例えば C12141→C16293 の予測代謝経路の場合、既知の代謝経路では、C16293 は以下の代謝経路により生合成される。

C08604－(C 環が付加)→C08620－(D,E 環が付加)→C16292－(-CH3 が付加)→C16293

一方、予測した代謝経路は以下の別ルートの代謝経路を表現している。

C08604－(-CH3 が付加)→C12141－(C,D,E 環が付加)→C16293

既知の代謝反応において中間体となる C08620 や C16292 が存在することから、元の経路の方が尤もらしいと言えるが、予測した代謝経路の様に、付加の順番が変わることで、網目状の代謝経路を構成している例は実際に存在する。C12141 と C16293 の構造は大きくかけ離れているが、類似係数を用いず、包含関係に着目することでこの様な代謝経路の予測も可能となっている。

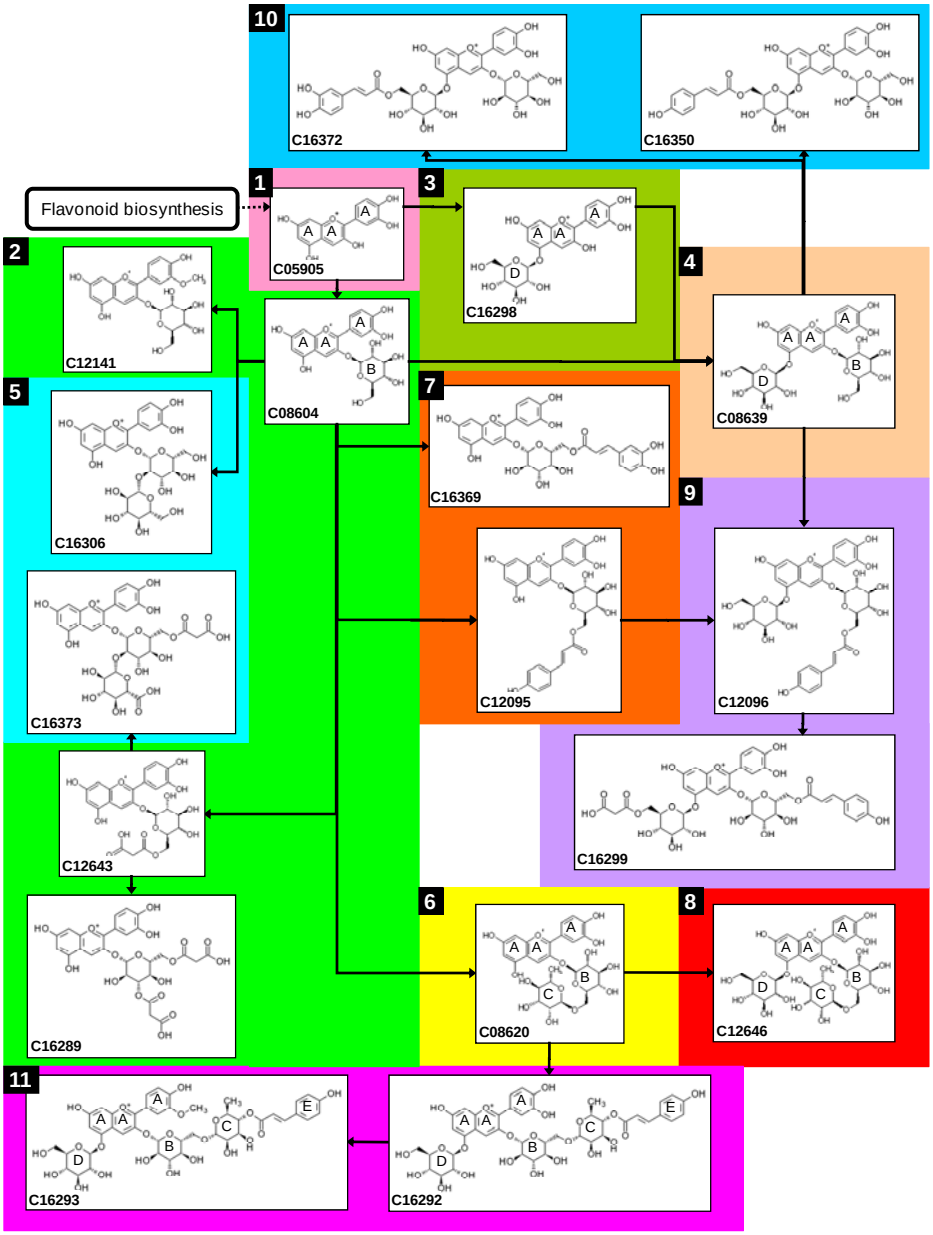


図 62 KEGG の代謝マップ map00942 の一部

図 56 を Compound レベル代謝経路予測結果と比較する為に、配置のみ変更した。

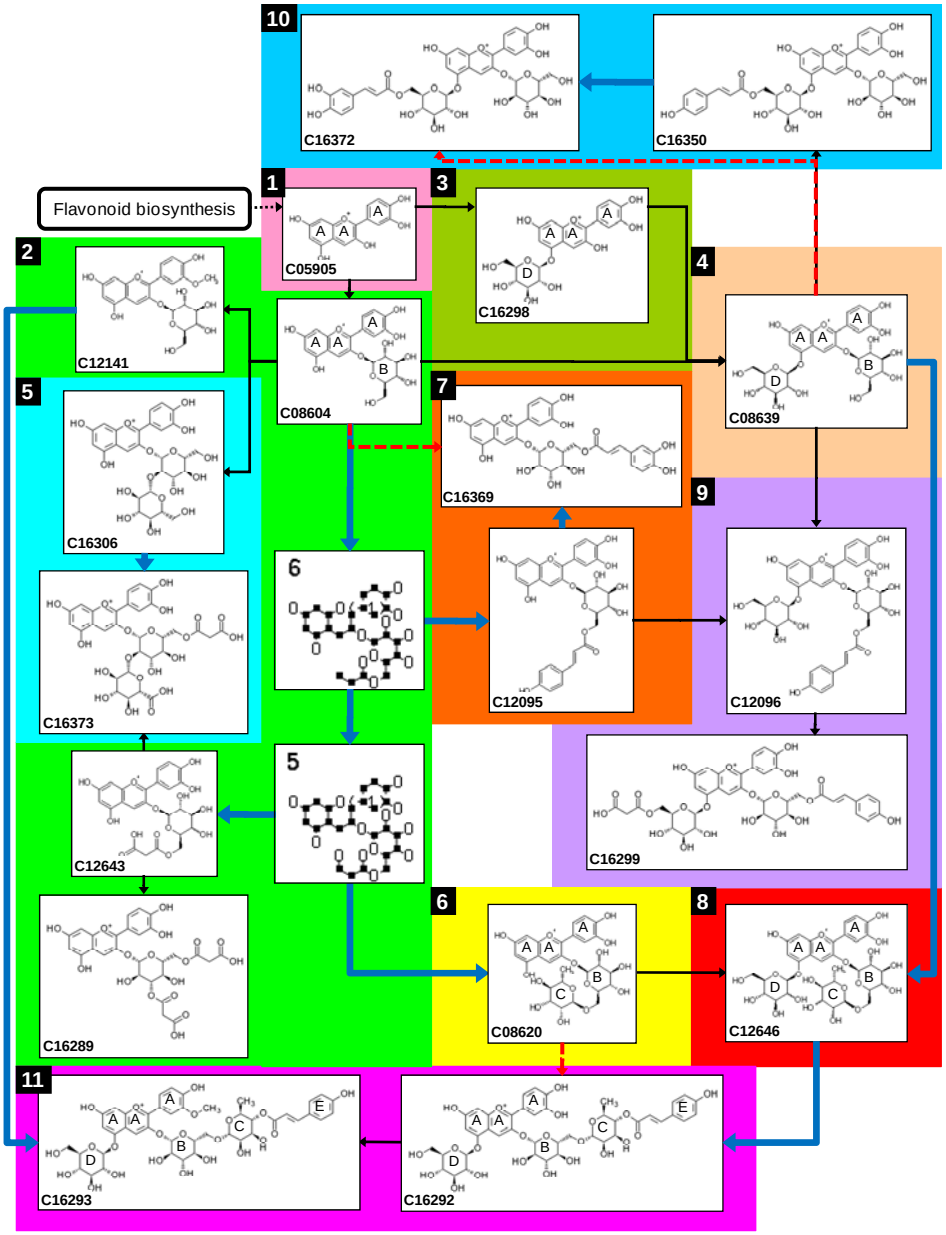


図 63 Compound レベル代謝経路予測結果

赤い破線が予測できなかった経路、青い太線が新しく予測した経路、5 と 6 の構造は中間構造として見つかった Subgraph を表している。

(1)や(2)で棄却された代謝経路の検出を目的として、2 つの概念「Direct pathway」および「Shortest pathway」を定義する。また、「Direct pathway」および「Shortest pathway」の例を図 64 に示す。

[定義 14] Direct pathway

中間構造のみを経由することを許した代謝物間の最短経路

[定義 15] Shortest pathway

他の代謝物の経由を許した代謝物間の最短経路

本研究で開発した MetClassifier(第 III 部第 11 章)では、Direct pathway および Shortest pathway に関する(1)～(4)の項目をタブ区切りのテキストファイルとして出力可能である。図 63 の予測代謝経路からは、図 65 の Direct pathway および図 66 の Shortest pathway が取得できる。

(1) 化合物 1 の Compound ID
(2) 化合物 2 の Compound ID
(3) 最短経路長
(4) 最短経路に出現する Subgraph ID（化合物 1-中間構造-化合物 2）
※Chemical graph と同形のとれている Subgraph ID には( )が付く。

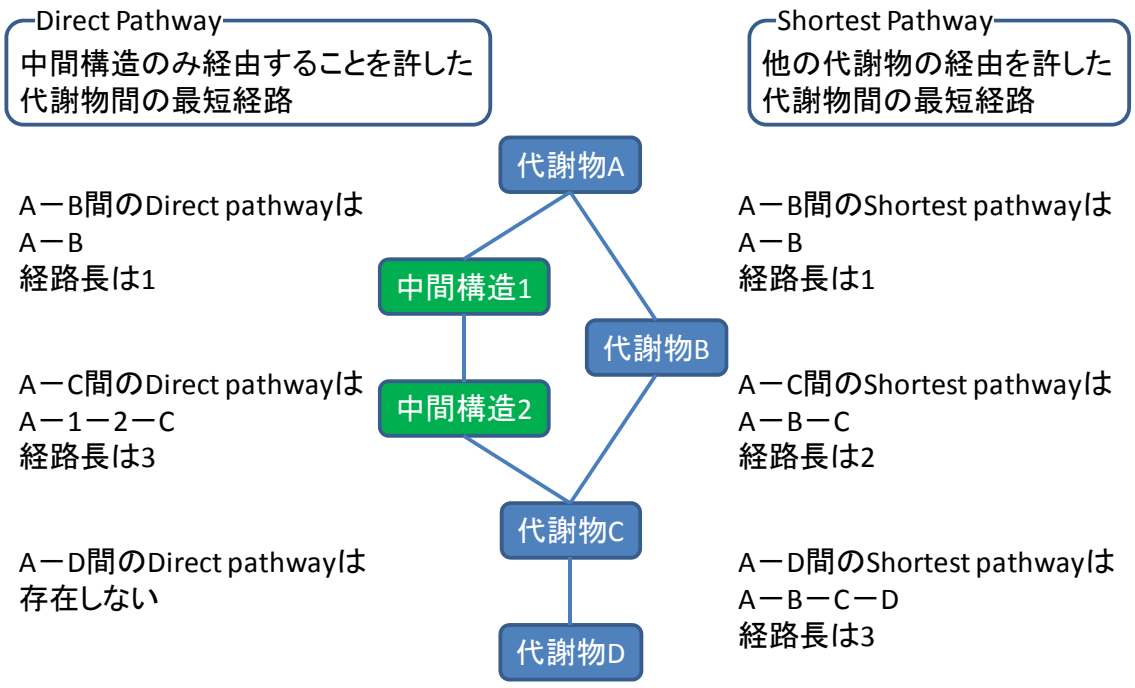


図 64 Direct pathway および Shortest pathway

C05905	C08604	1 (1)-(2)
C05905	C16298	1 (1)-(7)
C08604	C12141	1 (2)-(3)
C08604	C08639	1 (2)-(8)
C08604	C16306	1 (2)-(9)
C08604	C12095	2 (2)-6-(11)
C08604	C12643	3 (2)-6-5-(4)
C08604	C08620	3 (2)-6-5-(10)
C12141	C16293	1 (3)-(21)
C12643	C16289	1 (4)-(16)
C12643	C16373	1 (4)-(17)
C12643	C08620	2 (4)-5-(10)
C12643	C12095	3 (4)-5-6-(11)
C16298	C08639	1 (7)-(8)
C08639	C12646	1 (8)-(12)
C08639	C12096	1 (8)-(13)
C08639	C16350	1 (8)-(14)
C16306	C16373	1 (9)-(17)
C08620	C12646	1 (10)-(12)
C08620	C12095	3 (10)-5-6-(11)
C12095	C16369	1 (11)-(18)
C12095	C12096	1 (11)-(13)
C12646	C16292	1 (12)-(15)
C12096	C16299	1 (13)-(19)
C16350	C16372	1 (14)-(20)
C16292	C16293	1 (15)-(21)

図 65 図 63 の予測代謝経路から取得した Direct pathway

各列には以下の情報が出力される。

化合物 1 の Compound ID

化合物 2 の Compound ID

最短経路長

最短経路に出現する Subgraph ID（化合物 1-中間構造-化合物 2）

C05905 C08604 1	(1)-(2)
C05905 C16298 1	(1)-(7)
C05905 C12141 2	(1)-(2)-(3)
C05905 C08639 2	(1)-(2)-(8)
C05905 C16306 2	(1)-(2)-(9)
C05905 C16293 3	(1)-(2)-(3)-(21)
C05905 C12095 3	(1)-(2)-6-(11)
C05905 C12646 3	(1)-(2)-(8)-(12)
C05905 C12096 3	(1)-(2)-(8)-(13)
C05905 C16350 3	(1)-(2)-(8)-(14)
C05905 C16373 3	(1)-(2)-(9)-(17)
C05905 C16292 4	(1)-(2)-(3)-(21)-(15)
C05905 C12643 4	(1)-(2)-6-5-(4)
C05905 C08620 4	(1)-(2)-6-5-(10)
C05905 C16369 4	(1)-(2)-6-(11)-(18)
C05905 C16299 4	(1)-(2)-(8)-(13)-(19)
C05905 C16372 4	(1)-(2)-(8)-(14)-(20)
C05905 C16289 5	(1)-(2)-6-5-(4)-(16)
C08604 C12141 1	(2)-(3)
C08604 C08639 1	(2)-(8)
C08604 C16306 1	(2)-(9)
C08604 C16298 2	(2)-(1)-(7)
C08604 C16293 2	(2)-(3)-(21)
・・・	

図 66 図 63 の予測代謝経路から取得した Shortest pathway の一部

データ形式は Direct pathway と同じである。Shortest pathway は予測代謝経路が連結である場合、入力とした全ての代謝経路の組合せが出力される。間に他の代謝物を挟んでいても経路長が短い化合物ペアは構造が類似していることを意味しているので、正解の代謝経路の可能性がある。

第5.1.3項　　大規模データに対する予測精度の検証

本項では第 5.1.1 項で用いた RPAIR データベースから得た 4,976 代謝物から構成される 6,131 代謝物ペアを用いて、大規模データに対する予測精度の検証を行う。4,976 個の代謝物から 35,633 の Framework-based MFCS が抽出できた。さらに、包含関係に着目した代謝経路予測を行ったところ 164,936 の Framework-based MFCS レベルの代謝経路予測結果を得ることが出来た。第 5.1.2 項では、入力とした 19 個の Chemical graph から 21 個の Framework-based MFCS を抽出した。抽出した Framework-based MFCS の内、19 個が入力 Chemical graph と同形で、残りの 2 個が中間 Subgraph として抽出された。一方で、今回のケースでは、4,976 個の Chemical graph から、35,633 個の Framework-based MFCS を抽出している。これは、概算で入力 Chemical graph の約 7 倍に相当する中間 Subgraph を抽出したことになる。この様な膨大な数の中間 Subgraph が抽出された原因は、第 5.1.2 項で入力とした Chemical graph は Flavonoid に関する類似 Chemical graph であったのに対し、今回のケースでは RPAIR 全体から抽出している為、あらゆる種類の Chemical graph が含まれていることが原因である。多様性のある Chemical graph に対して、Framework-based MFCS の抽出を試みた場合、類似化合物間でクラスタが作成され、クラスタ間に大量の中間 Subgraph が生成される(図 67)。

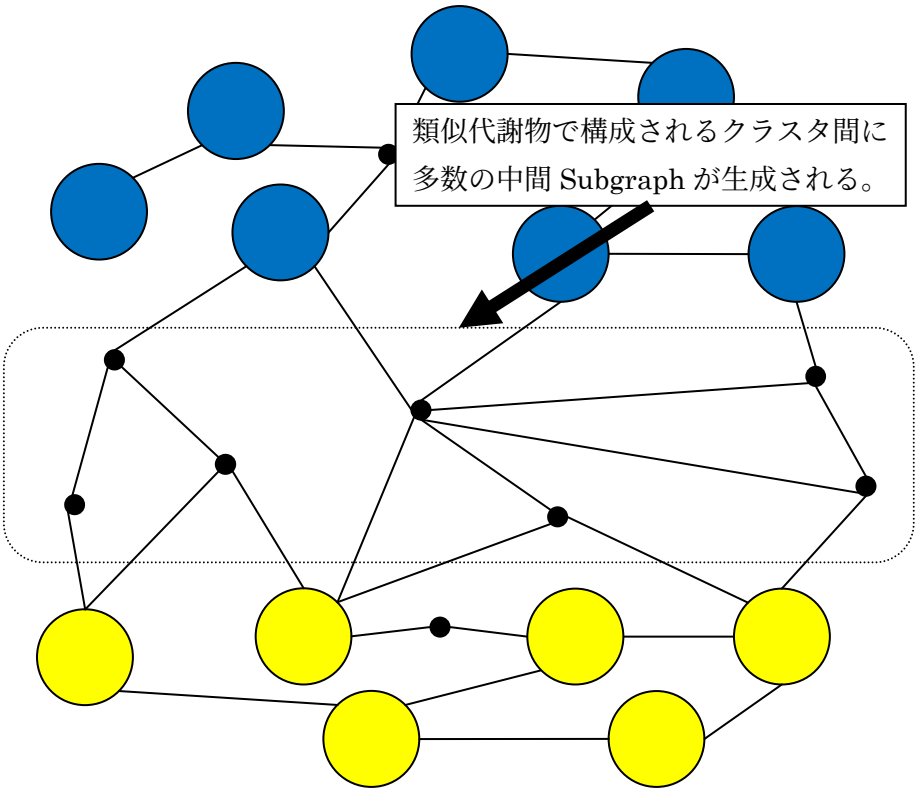


図 67 多様性のある Chemical graph のセットに対して提案手法を適用した場合
類似化合物間でクラスタが形成され、クラスタ間に多数の中間 Subgraph が生成される。

中間 Subgraph の増加に伴い、Framework-based MFCS レベルの代謝経路予測結果の数も増大する。そこで、Direct pathway および Shortest pathway を用いることで有意な代謝経路の抽出を試みた。

表 2 は予測代謝経路上における、代謝物ペア間の Direct pathway および Shortest pathway の経路長の分布を示している。1 列目は経路長、2,3 列目は全代謝物ペア間の経路長の分布、4,5 列目は RPAIR に含まれる代謝物ペア間の経路長の分布、6,7 列目は RPAIR に含まれない代謝物ペア間の経路長の分布をそれぞれ表している。2 列目=4 列目+6 列目、3 列目=5 列目+7 列目である。

表 2 Direct pathway および Shortest pathway の経路長の分布

Path length	All predicted pathway		Correct pathway		New predicted pathway	
	Direct	Shortest	Direct	Shortest	Direct	Shortest
0	2,569	2,569	740	740	1,829	1,829
1	12,880	12,880	2,151	2,151	10,729	10,729
2	37,492	63,397	798	1,252	36,694	62,145
3	156,098	337,751	373	731	155,725	337,020
4	621,540	1,363,657	425	416	621,115	1,363,241
5	1,299,755	2,724,546	374	366	1,299,381	2,724,180
6	1,855,679	3,198,222	307	237	1,855,372	3,197,985
7	2,020,816	2,455,855	266	140	2,020,550	2,455,715
8	1,756,446	1,346,170	169	71	1,756,277	1,346,099
9	1,273,612	589,243	101	21	1,273,511	589,222
10	830,142	207,921	117	6	830,025	207,915
11	500,419	58,786	84		500,335	58,786
12	281,622	13,694	25		281,597	13,694
13	146,517	2,575	11		146,506	2,575
14	77,787	465	7		77,780	465
15	42,251	66	4		42,247	66
16	20,869	3			20,869	3
17	9,578		3		9,575	
18	4,645				4,645	
19	2,433				2,433	
20	1,110				1,110	
21	384				384	
22	142				142	
23	42				42	
24	20				20	
25	2				2	
Sum	10,954,850	12,377,800	5,955	6,131	10,948,895	12,371,669

まず、経路長=0 は代謝物ペアが同形であったことを意味している。提案手法では多重結合や立体異性に関連する情報は予測の際に考慮に入れていない為、異性化反応や結合のタイプが変わる反応では代謝物ペア間の経路長=0 となる。経路長=1 は中間構造が存在しない最も化学構造が近い代謝物ペアが該当する。経路長が 0 もしくは 1 の代謝物ペアは、代謝経路が既知の化合物ペア全体の 47%(2,891/6,131)に相当する。この結果は、既知の代謝反

応の多くが提案手法によって正しく予測されたことを意味している。また、6,7 列目の RPAIR に含まれない代謝物ペアの内、経路長が 0 もしくは 1 である(1,829+10,729=)12,558 代謝物ペアに関しても、新規代謝経路である可能性が高い。

代謝経路が既知の代謝物ペア A, B の中間代謝物もしくは中間 Subgraph が存在した場合に、予測代謝経路上において A, B は隣り合わなくなる(第 5.1.2 項での考察)。この為、代謝経路予測を行った後、経路長が 0 や 1 の代謝物ペアだけではなく、経路長が 2 以上の代謝物ペアも実在する代謝経路の可能性がある。ここで、RPAIR に含まれる化合物ペア間の経路長の分布に着目してみると、4 列目の Direct pathway ケースでは、経路長=0 となるのは 12.4%(740/5,955)、経路長=1 となるのは 36.1%(2,151/5,955)、経路長=2 となるのは 13.4%(798/5,955)となり、経路長≦2 となるのが 61.9%を占めている。5 列目の Shortest pathway ケースでは、経路長=0 となるのは 12%(740/6,131)、経路長=1 となるのは 35%(2,151/6,131)、経路長=2 となるのは 20%(1,252/6,131)となり、経路長≦2 となるのが 67%を占めている。6,7 列目の RPAIR に含まれない代謝物ペアの中にも同じ割合で実在する代謝経路が含まれている可能性があり、これらの指標が利用できる。

既知の代謝反応であっても経路長が長く予測されるケースも存在する。例えば、RPAIR に含まれる代謝物ペア間の Direct pathway の項目で経路長が 17 の代謝物ペアに着目してみる。該当する代謝物ペアには、RP04551(C00065-C05699)、RP04535 (C00109-C05699)、RP12437 (C05702-C05699)が含まれており、いずれも C05699 の代謝に関係していた。そこで、C05699 に関連する Direct pathway と Shortest pathway について詳しく調べてみたところ、図 68 および図 69 の結果を得た。例えば RP04551 (C00065-C05699)に対して、提案手法による代謝経路予測を行った場合、中間 Subgraph として、入力 Chemical graph と同形な Framework-based MFCS C00041(37)、C02432(5621)、C05689(12941)を検出し、図 70 に示す伸長反応を予測している。この為、RP04551(C00065-C05699)は予測代謝経路上で隣り合うことが出来ず、入力 Chemical graph と同形のとれなかった 15 個の Framework-based MFCS を経由した、経路長が長い Direct pathway が出力される(図 68)。

```
C00022  C05699  15      (11)-7491-7492-7494-7496-7497-5-2-12940-12950-5623-5622-12942-12947-12946-(34665)
C00065  C05699  17      (500)-501-38-455-456-467-468-1273-476-2-12940-12950-5623-5622-12942-12947-12946-(34665)
C00109  C05699  17      (60)-64-68-48-51-7043-5589-4-5-2-12940-12950-5623-5622-12942-12947-12946-(34665)
C05688  C05699  5       (5621)-5622-12942-12947-12946-(34665)
C05698  C05699  6       (11792)-5623-5622-12942-12947-12946-(34665)
C05702  C05699  17      (3153)-3154-3155-3157-3158-547-533-534-5-2-12940-12950-5623-5622-12942-12947-12946-(34665)
```

図 68 C05699 に関連する Direct pathway

```
C00022  C05699  6       (11)-(14)-(37)-(5621)-(12941)-12946-(34665)
C00065  C05699  5       (500)-(37)-(5621)-(12941)-12946-(34665)
C00109  C05699  6       (60)-(43)-(28)-(11792)-(11791)-12945-(34665)
C05688  C05699  3       (5621)-(12941)-12946-(34665)
C05698  C05699  3       (11792)-(11791)-12945-(34665)
C05702  C05699  6       (3153)-(95)-(28)-(11792)-(11791)-12945-(34665)
```

図 69 C05699 に関連する Shortest pathway

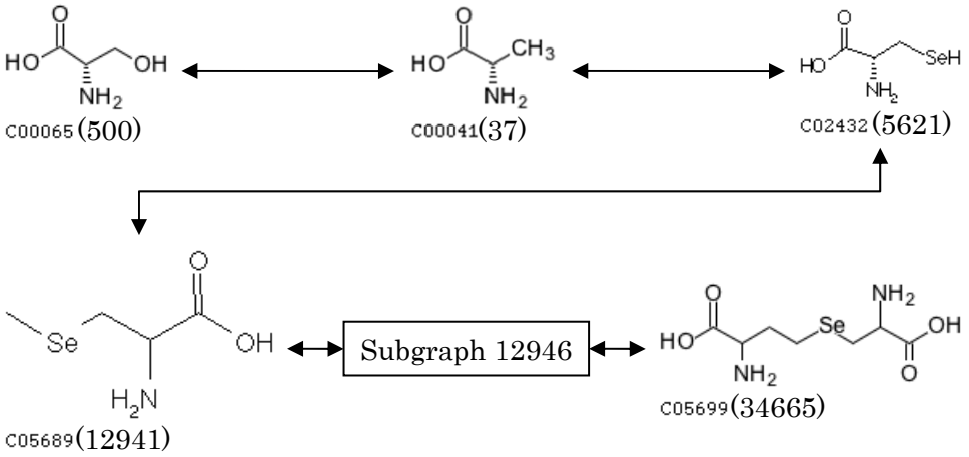


図 70 予測代謝経路上における C00065-C05699 間の Shortest pathway

一方で、C05699 に関連する RPAIR の根拠となっている既知の代謝反応は図 71～図 76 の 6 反応である。例えば図 73 の反応は、C00065 と C05698 の脱水反応により C05699 が得られる反応である。これを根拠にして、C00065-C05699 および C05698-C05699 はいずれも RPAIR において main と定義される。提案手法では、化学構造に類似性の認められる代謝物ペアを代謝経路として予測している為、複数の基質から生成物が作られるような反応の場合、各基質単体と生成物の類似性は小さくなる為、これらを予測することは難しい。

一方で、予測代謝経路上における Direct pathway もしくは Shortest pathway の経路長(図 68、図 69)と、代謝反応の描画情報(図 71～図 76)を比較してみると、経路長の短い(C05688-C05699)、(C05698-C05699)に関しては、図 71～図 76 の中で左右に配置されており、経路長の長い(C00022-C05699)、(C00065-C05699)、(C00109-C05699)、(C05702-C05699)に関しては、いずれも図 71～図 76 の中で下部に配置されている。RPAIR において同じ main として登録されている代謝物ペアの中にも化学構造の類似しているペアと類似していないペアが存在し、基質と生成物のペアで最も類似している代謝物が左右に配置されていることが観察できる。提案手法による代謝経路予測では、構造の類似している代謝物ペアに関しては中間構造が存在しにくくなる為、経路長は短くなり、構造が類似していない代謝物ペアに関しては、考えられる中間構造の数が増える為、経路長は長くなる。つまり、経路長は構造の類似性を表現しており、既知の代謝反応の描画情報とも一致しており、この点に関しては代謝反応の描画を行った専門家の意見と同じ判断が出来ていることを意味している。

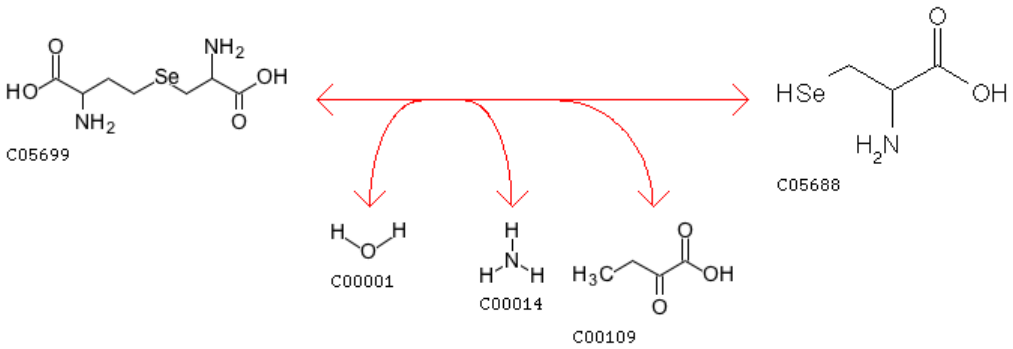


図 71 C05699 に関連する代謝反応(R04930)

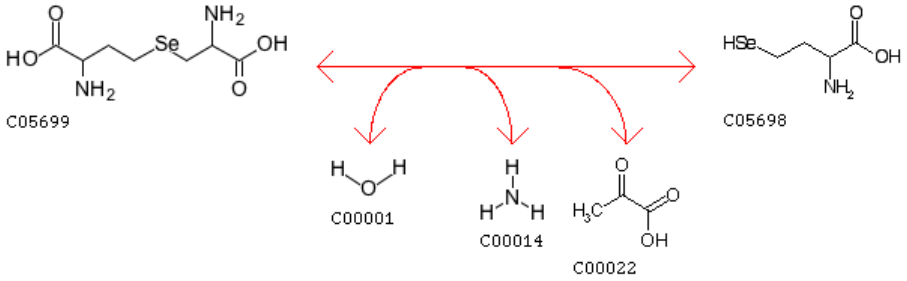


図 72 C05699 に関連する代謝反応(R04941)

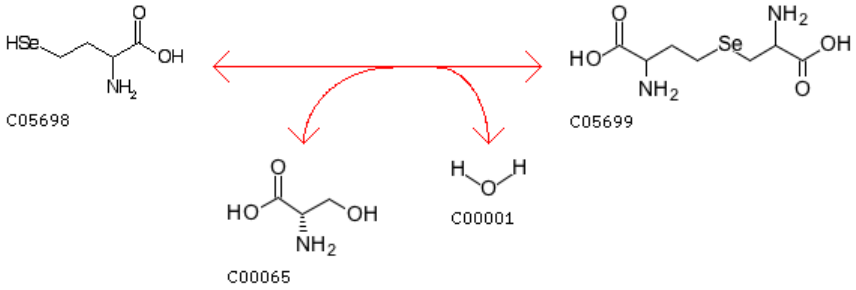


図 73 C05699 に関連する代謝反応(R04942)

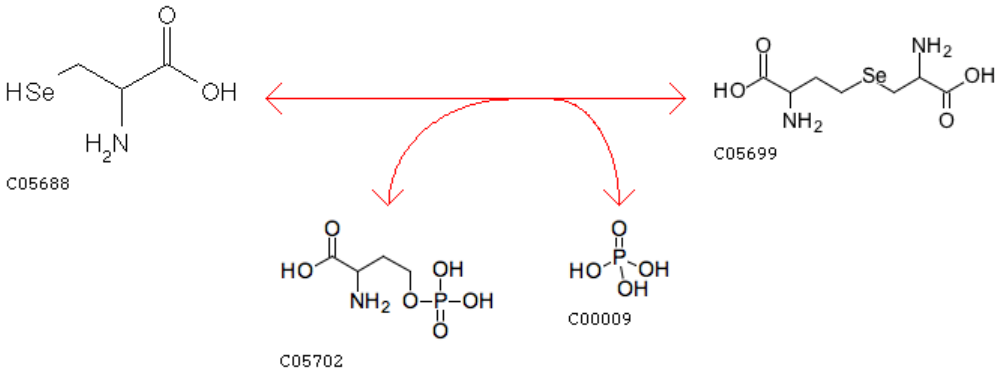


図 74 C05699 に関連する代謝反応(R04944)

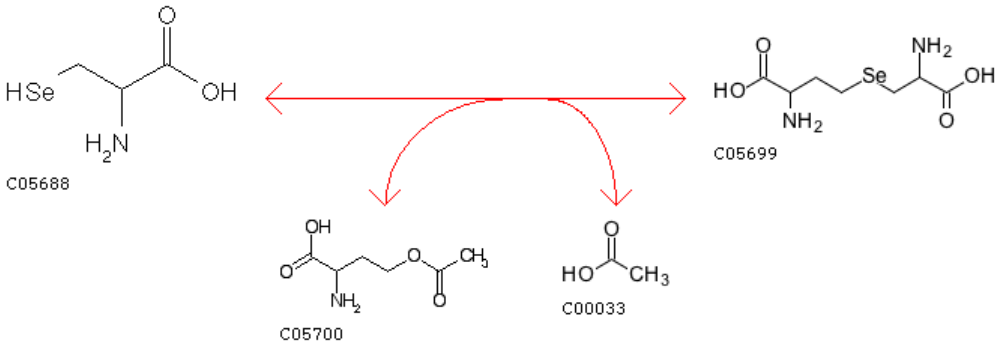


図 75 C05699 に関連する代謝反応(R04945)

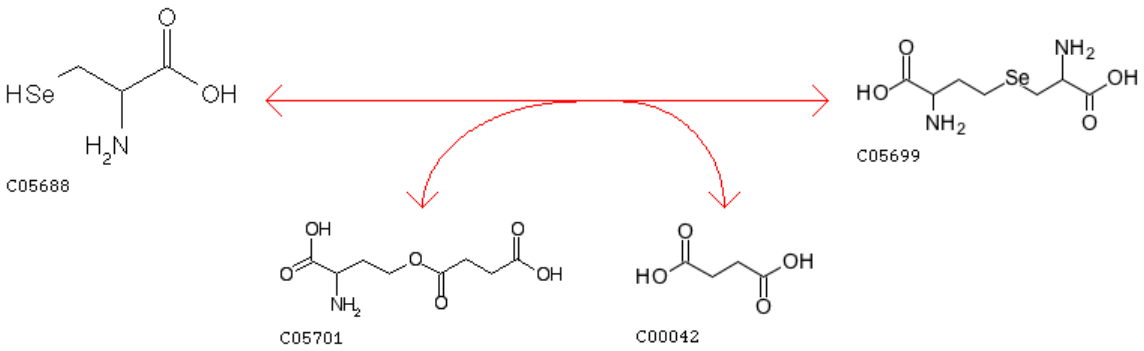


図 76 C05699 に関連する代謝反応(R04946)

RPAIR に含まれる代謝物ペアの内、予測代謝経路上における Direct pathway と Shortest pathway の経路長が異なる代謝物ペアは、中間構造となる別の代謝物が発見された代謝物ペアに相当し、構造の類似性は低いが代謝反応が存在する代謝物ペアであることを意味している。このようなケースでは、既知の代謝反応とは異なる別の代謝反応を提案手法が予測している為、確認する価値がある。Direct pathway と Shortest pathway の経路長が異なる代謝物ペアを調査し整理した結果が表 3 であり、1,756 代謝物ペアが該当する。

提案手法を用いず、マニュアルオペレーションにより代謝経路を予測しようとした場合、4,976 代謝物から得られる代謝物ペアは 4,976×4,975÷2＝12,377,800 存在し、この中から有意な代謝経路を見つけ出す作業は非常に困難である。これに対し、提案手法を用いることで 12,558+1,756=14,314 の有意な代謝物ペアを提示できている。このことは、代謝経路を整理する際、作業量の大幅な低減に繋がることを意味している。

表 3 Direct pathway と Shortert pathway の異なる化合物ペアの分布

Path length	Correct pathway		Comparison of correct pathway length		
	Direct	Shortest	Same	Direct	Shortest
0	740	740	740	–	0
1	2,151	2,151	2,151	0	0
2	798	1,252	798	0	454
3	373	731	349	24	382
4	425	416	156	269	260
5	374	366	108	266	258
6	307	237	53	254	184
7	266	140	18	248	122
8	169	71	1	168	70
9	101	21	1	100	20
10	117	6	0	117	6
11	84		0	84	0
12	25		0	25	0
13	11		0	11	0
14	7		0	7	0
15	4		0	4	0
16			0	0	0
17	3		0	3	0
18			0	0	0
19			0	0	0
20			0	0	0
21			0	0	0
22			0	0	0
23			0	0	0
24			0	0	0
25			0	0	0
∞			0	176	0
Sum	5,955	6,131	4,375	1,756	1,756

第5.2節　　代謝経路が未知の代謝物を用いた提案手法の評価

第 5.1 節において、提案手法を用いることにより、多くの既知代謝経路を正しく予測できることが分かった。本節では、KNApSAcK データベースに登録されている 34,653 代謝物、19,293 生物種、75,054 代謝物-生物種関連情報を用いて提案手法を用いた代謝経路予測試みる。第 5.2.1 項では *Camellia sinensis*(お茶)に含まれる代謝物を対象として生物種固有の代謝経路予測を行う。第 5.2.2 項では、KNApSAcK に含まれる全 34,653 代謝物を用いて、代謝経路予測を行うことで、大規模データに対しても提案手法が動作することを検証する。

第5.2.1項　　生物種固有の代謝経路予測

本項では、KNApSAcK データベースを用い、*Camellia sinensis*(お茶)に含まれる代謝物を対象として、生物種固有の代謝経路予測を行う。KNApSAcK データベースに含まれる代謝物情報を代謝物－生物種関連情報を用いて *Camellia sinensis* の保有する代謝物に絞り込んだところ、123 代謝物に絞り込むことが出来た(図 77～図 80)。123 代謝物から 42 個のユニークな Framework を抽出することが出来た(図 81、図 82)。Framework レベルの代謝経路予測結果に着目してみると、*Camellia sinensis* には、主に、Flavonoid、Steroid、Gibberellin、に関する代謝物を含んでいることが確認できる(図 83)。

次に Compound レベルの代謝経路予測を行う。抽出した Framework-based MFCS を図 84～図 91 に示す。また、予測した代謝経路を図 92 に示す。Framework レベルの代謝経路予測結果と同様に、Flavonoid(図 93)、Steroid(図 94)、Gibberellin(図 95)、のクラスタを確認することが出来る。また、第 5.1.3 項でも議論にも出てきた中間 Subgraph で構成されるクラスタが確認できるが、Flavonoid、Steroid、Gibberellin のクラスタとは分離されていることが分かる。

第 5.1.2 項では、入力とした代謝物が Flavonoid に関連する代謝物のみであったが、化学構造に多様性のある代謝物をデータセットとして用いても、提案手法が有効であることが確認できた。

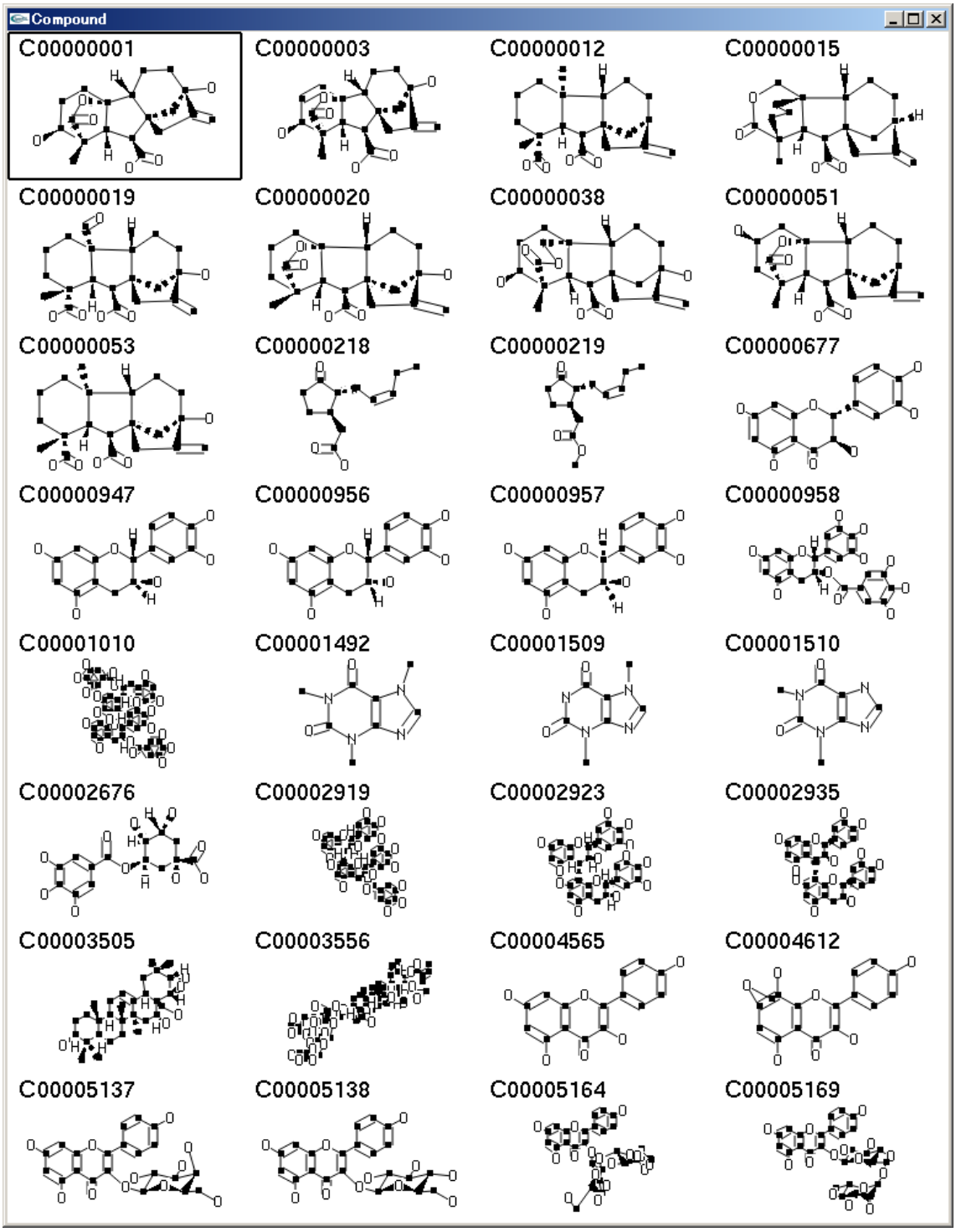


図 77 KNApSAcK に登録されている *Camellia sinensis* が保有する代謝物 1

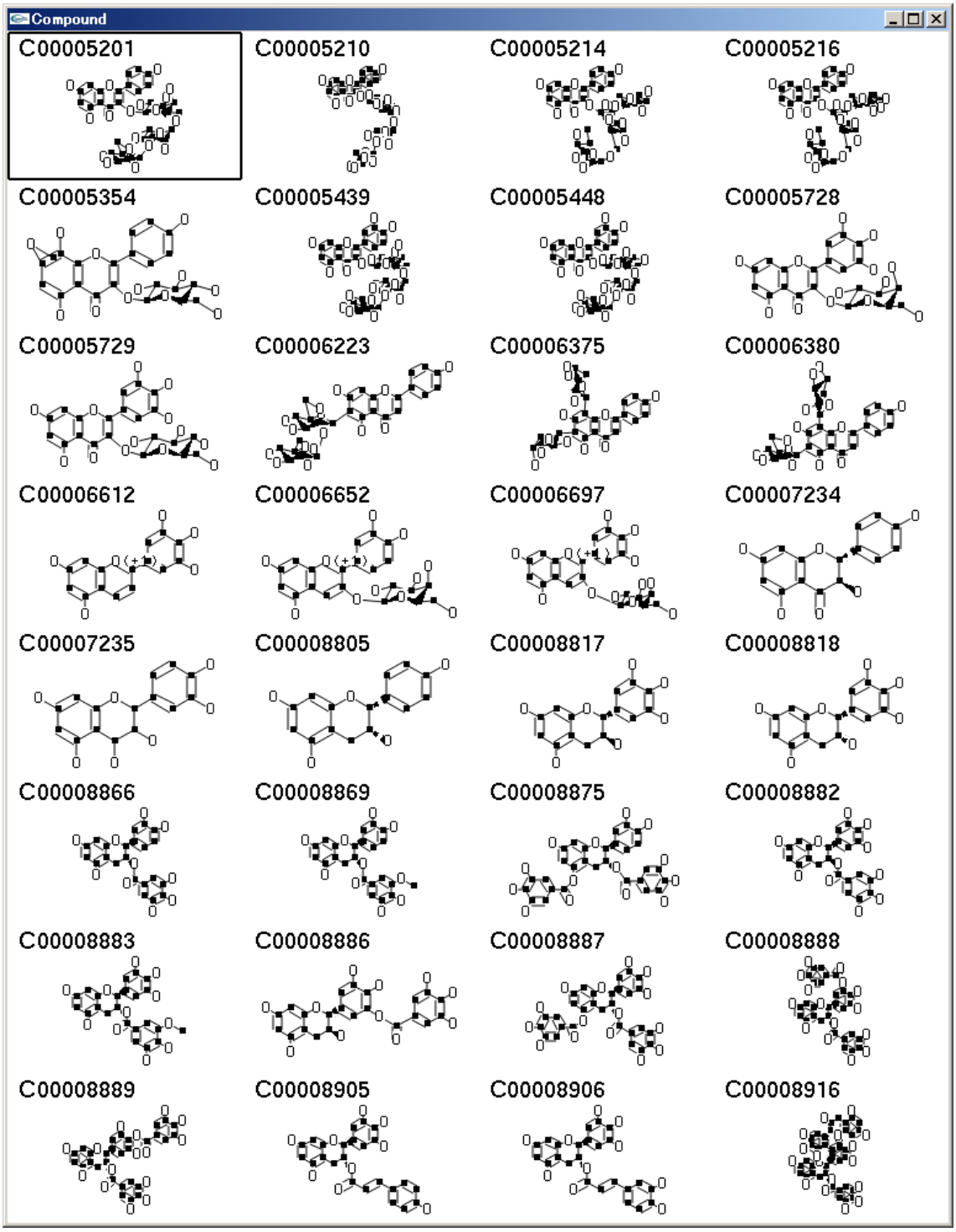


図 78 KNApSAcK に登録されている *Camellia sinensis* が保有する代謝物 2

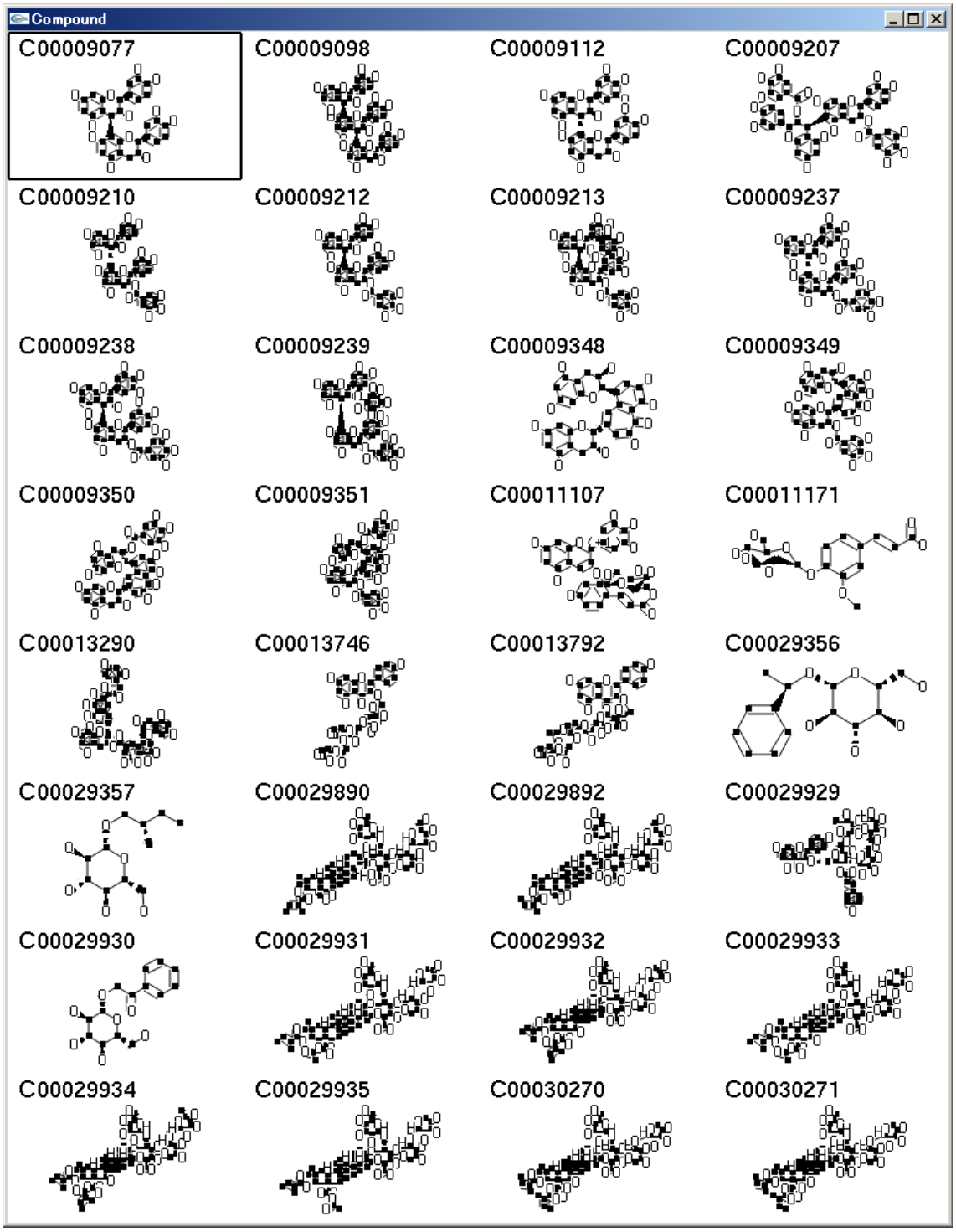


図 79 KNApSAcK に登録されている *Camellia sinensis* が保有する代謝物 3

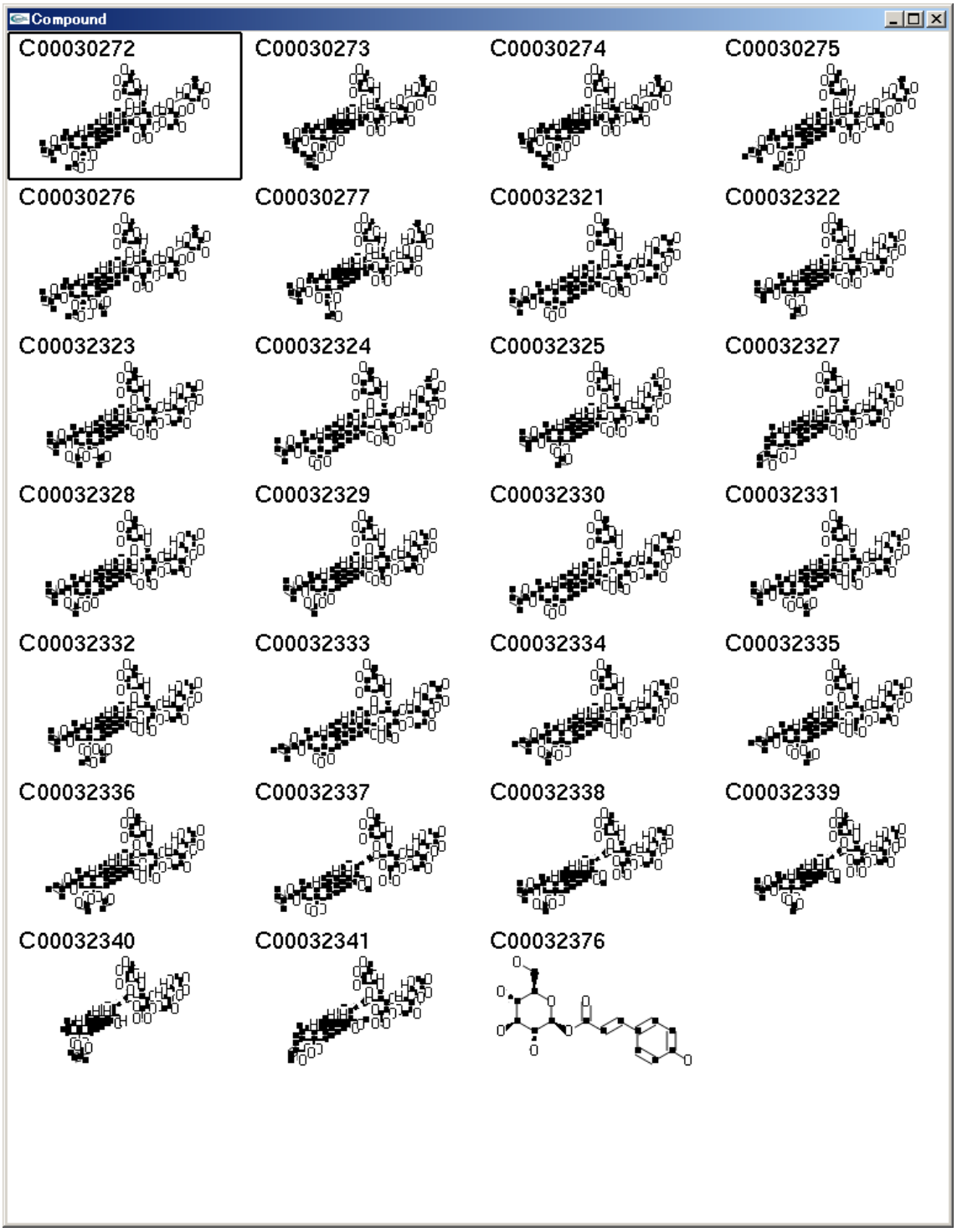


図 80 KNApSAcK に登録されている *Camellia sinensis* が保有する代謝物 4

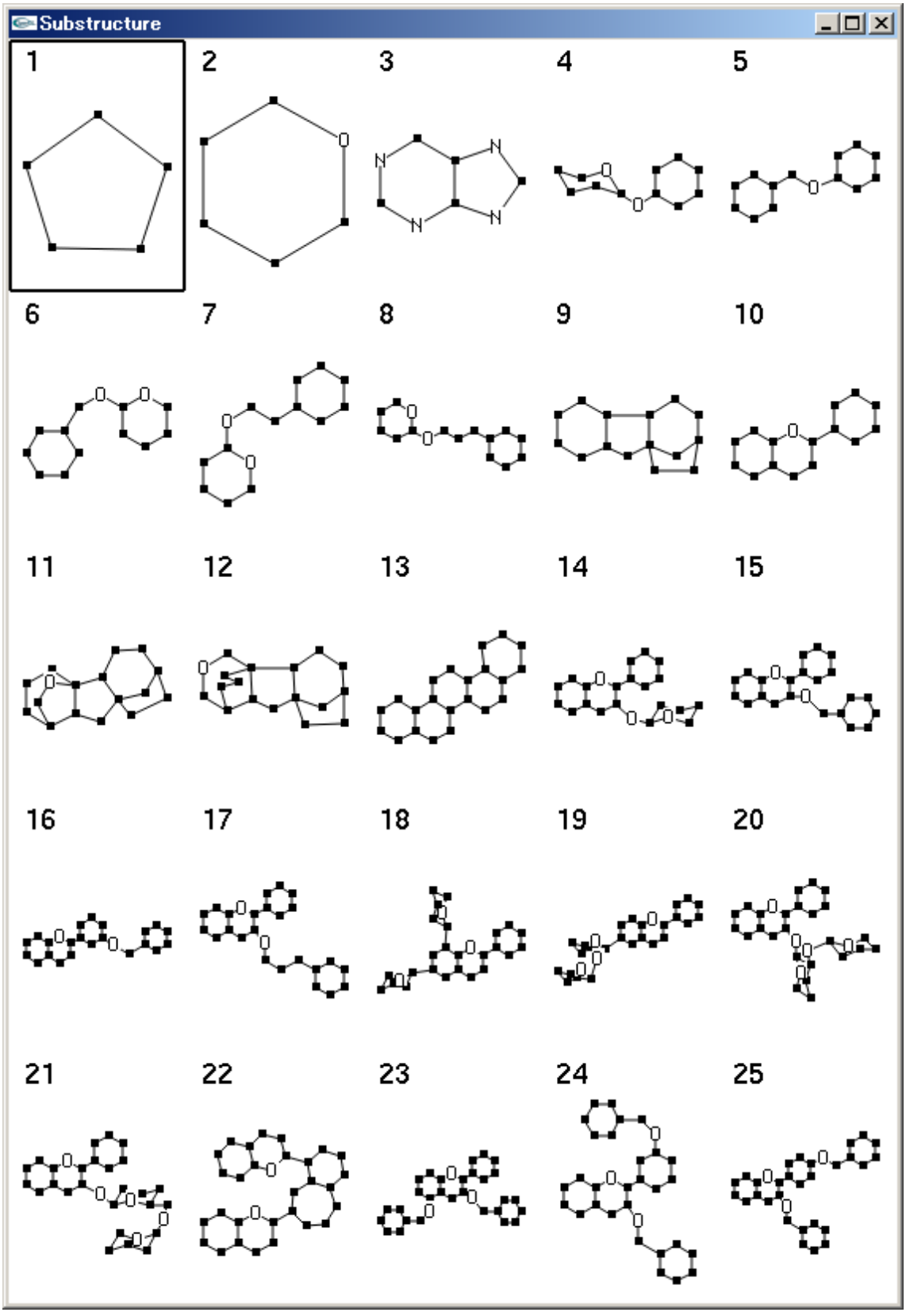


図 81 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework 1

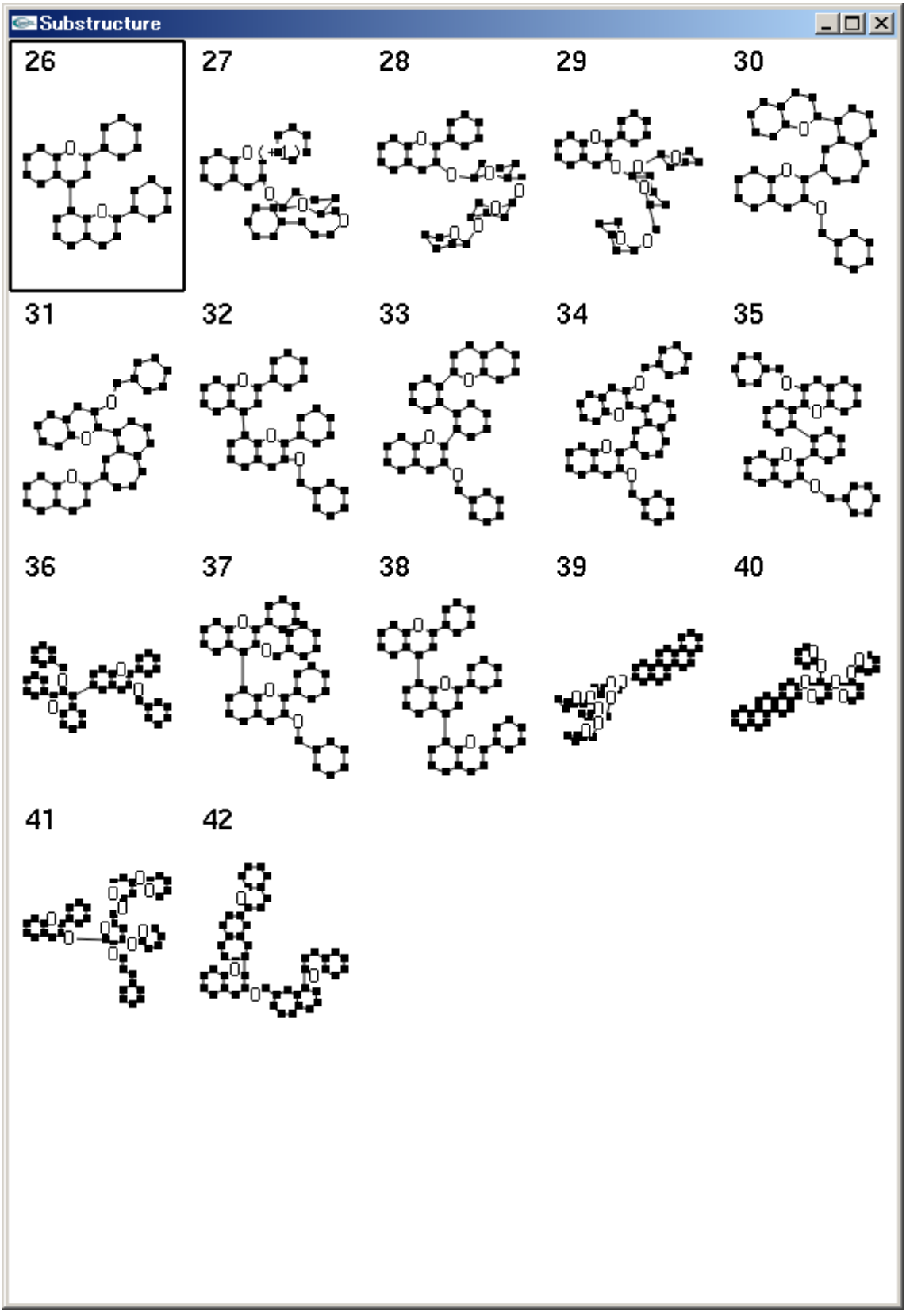


図 82 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework 2

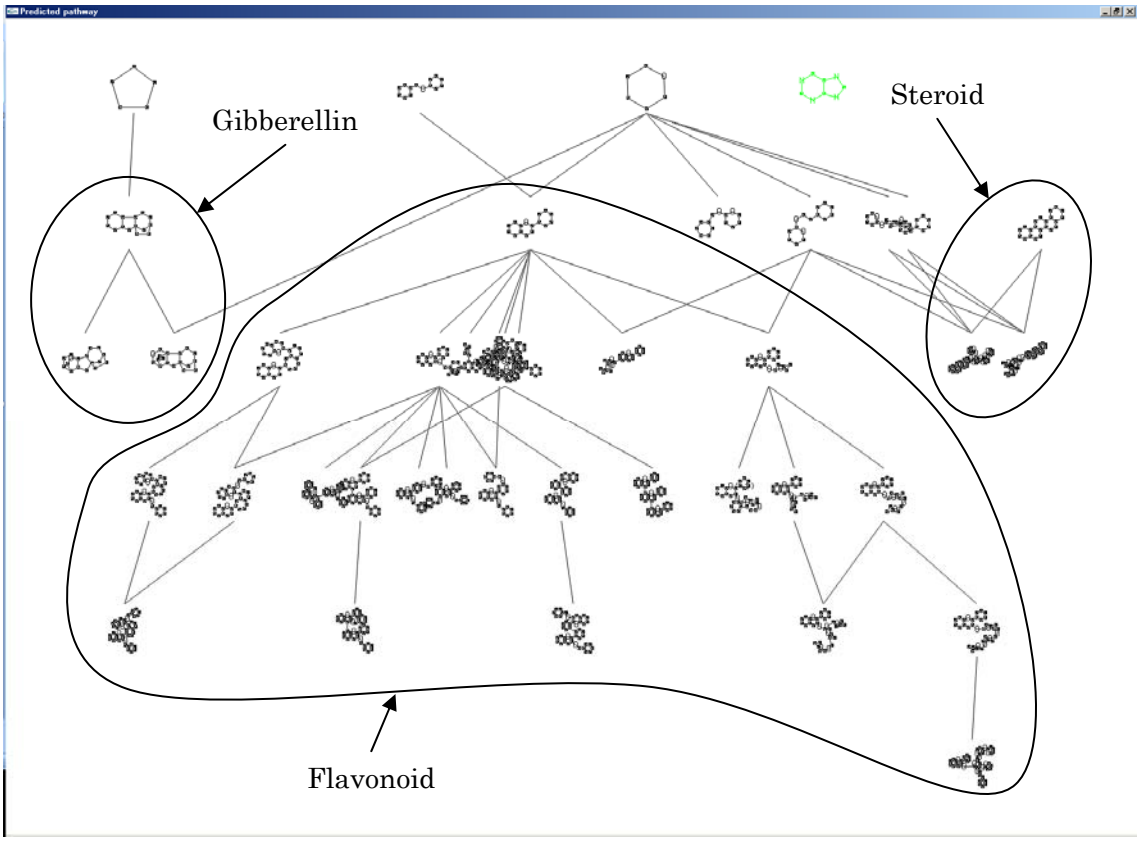


図 83 Framework レベルの代謝経路予測

Camellia sinensis には、主に、Flavonoid、Steroid、Gibberellin、に関する代謝物が報告されていることが分かる

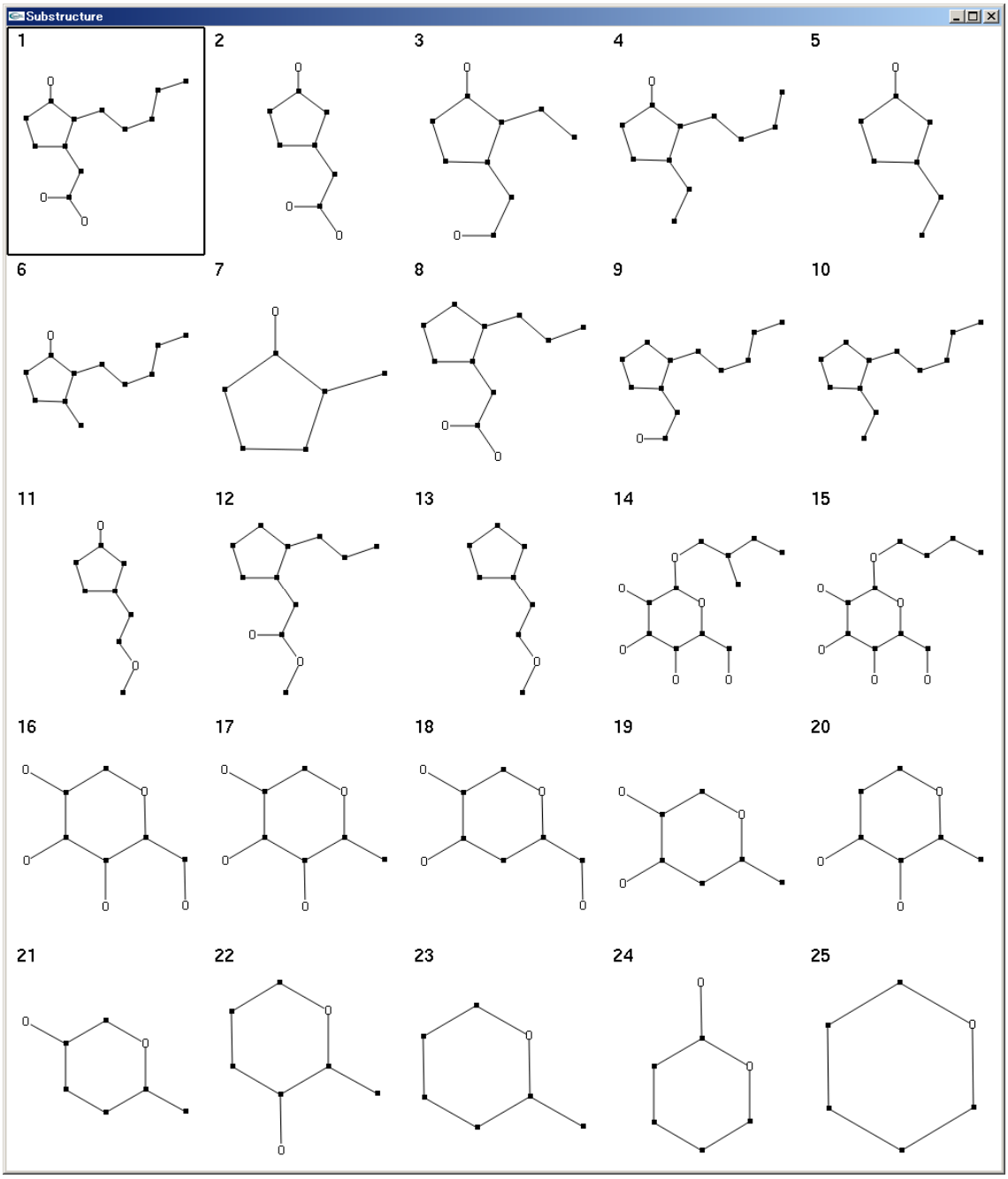


図 84 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 1

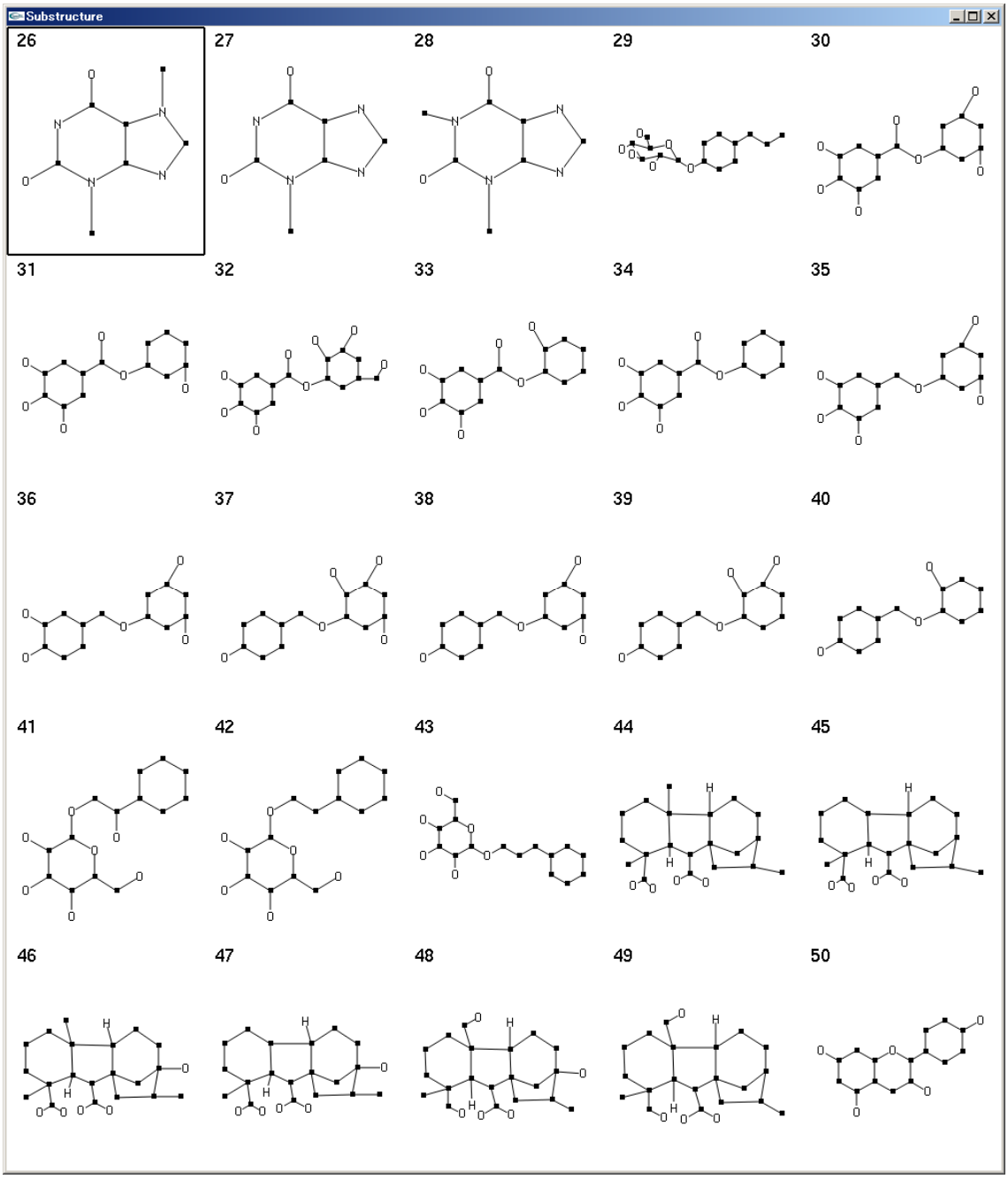


図 85 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 2

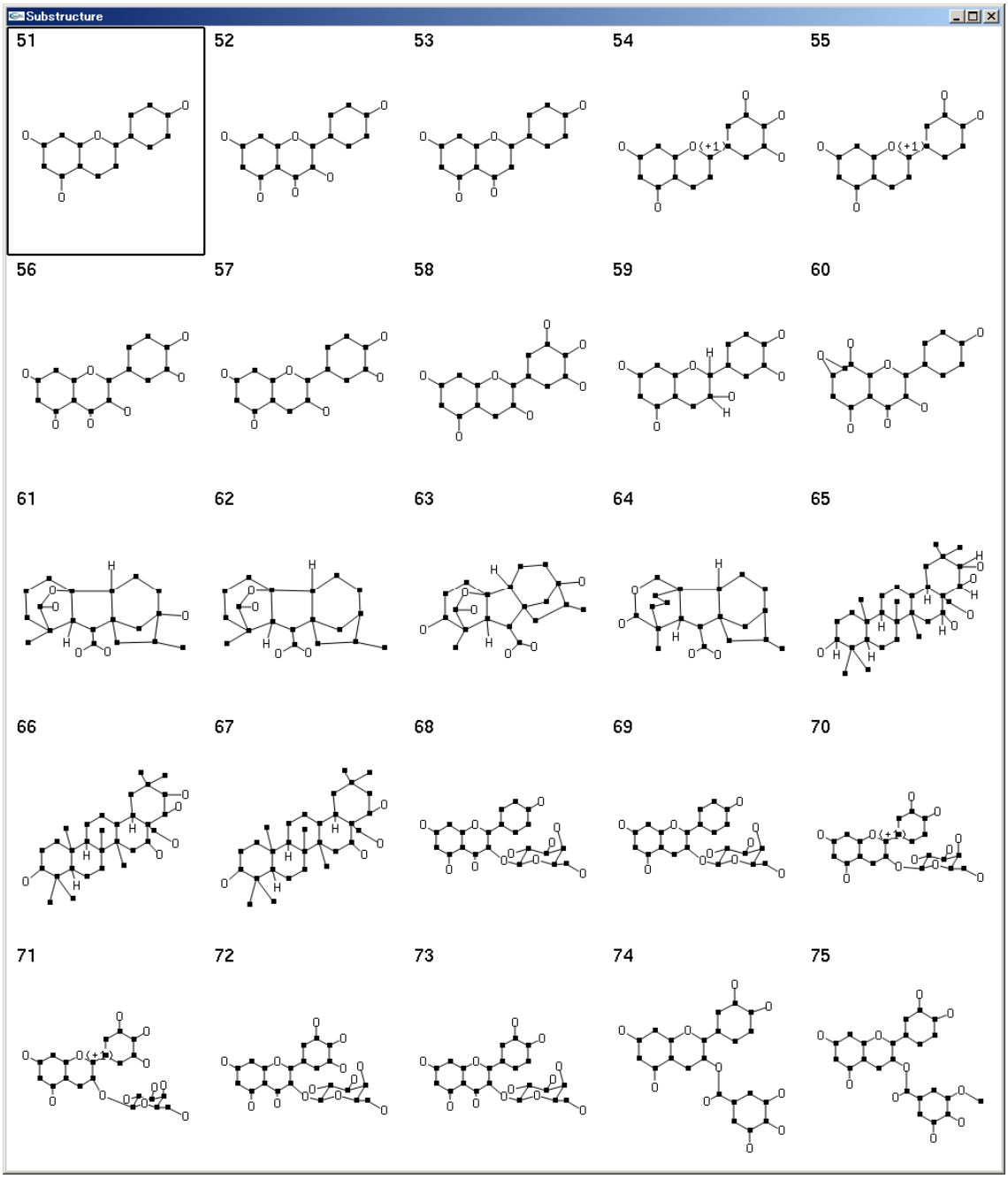


図 86 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 3

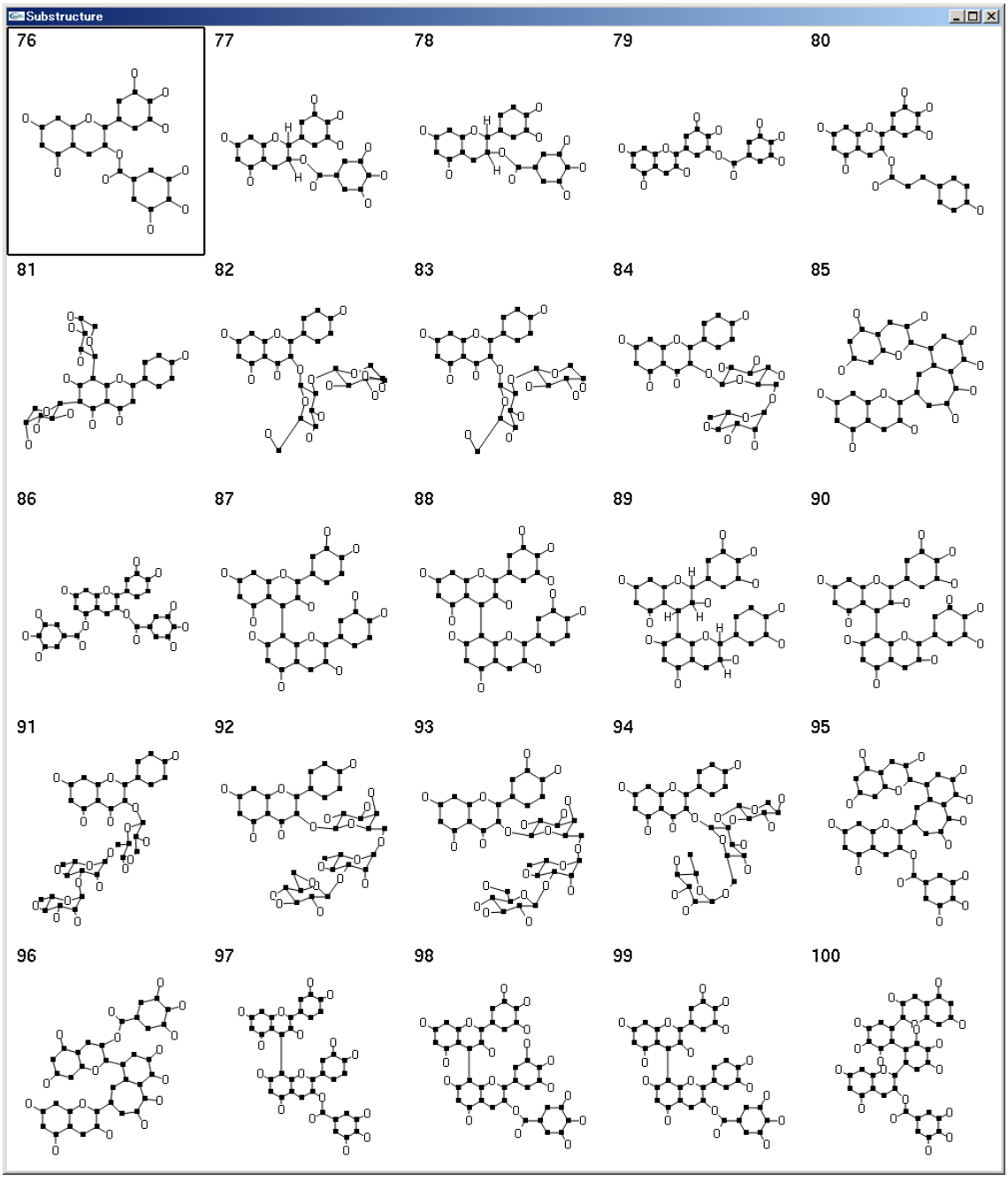


図 87 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 4

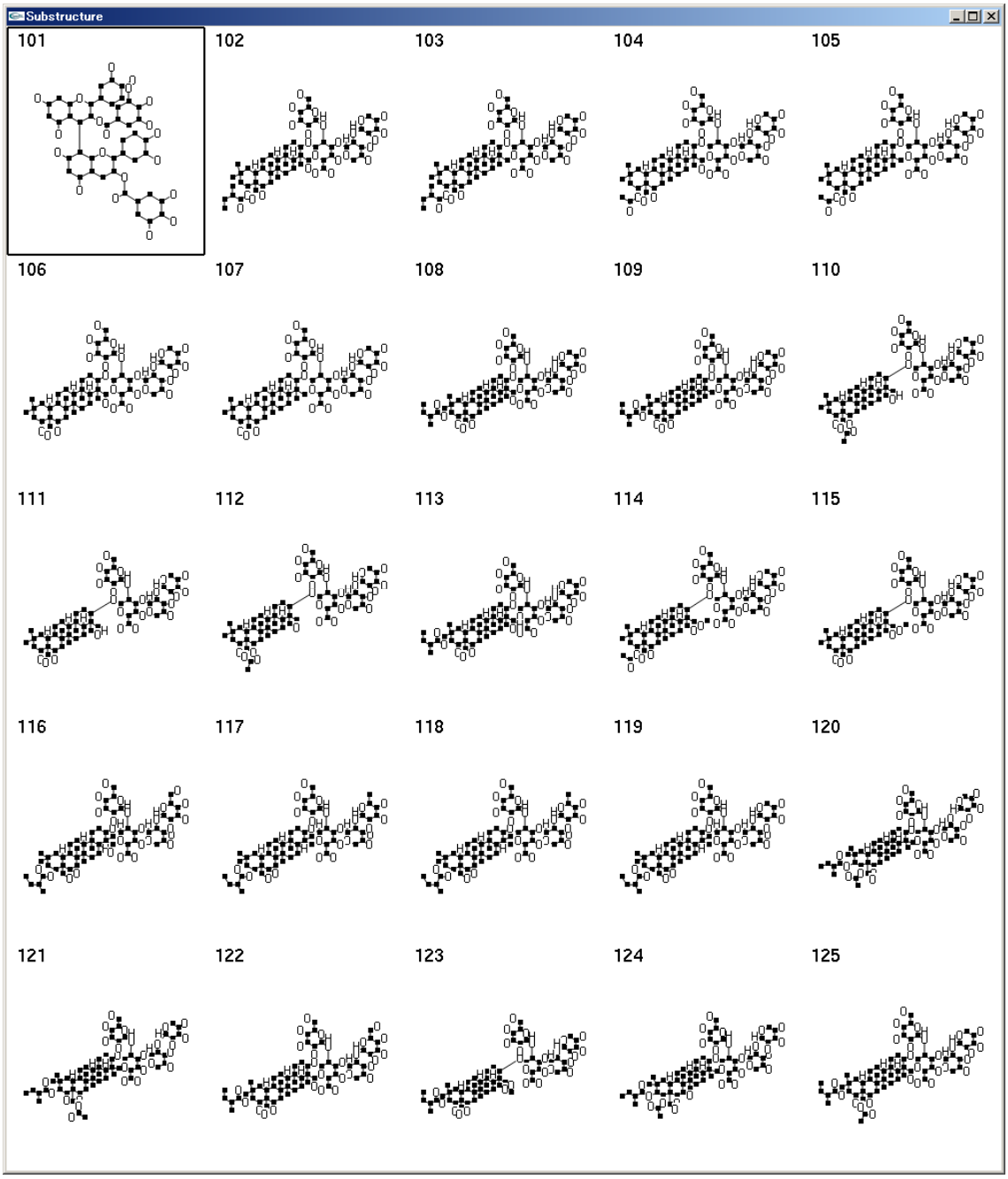


図 88 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 5

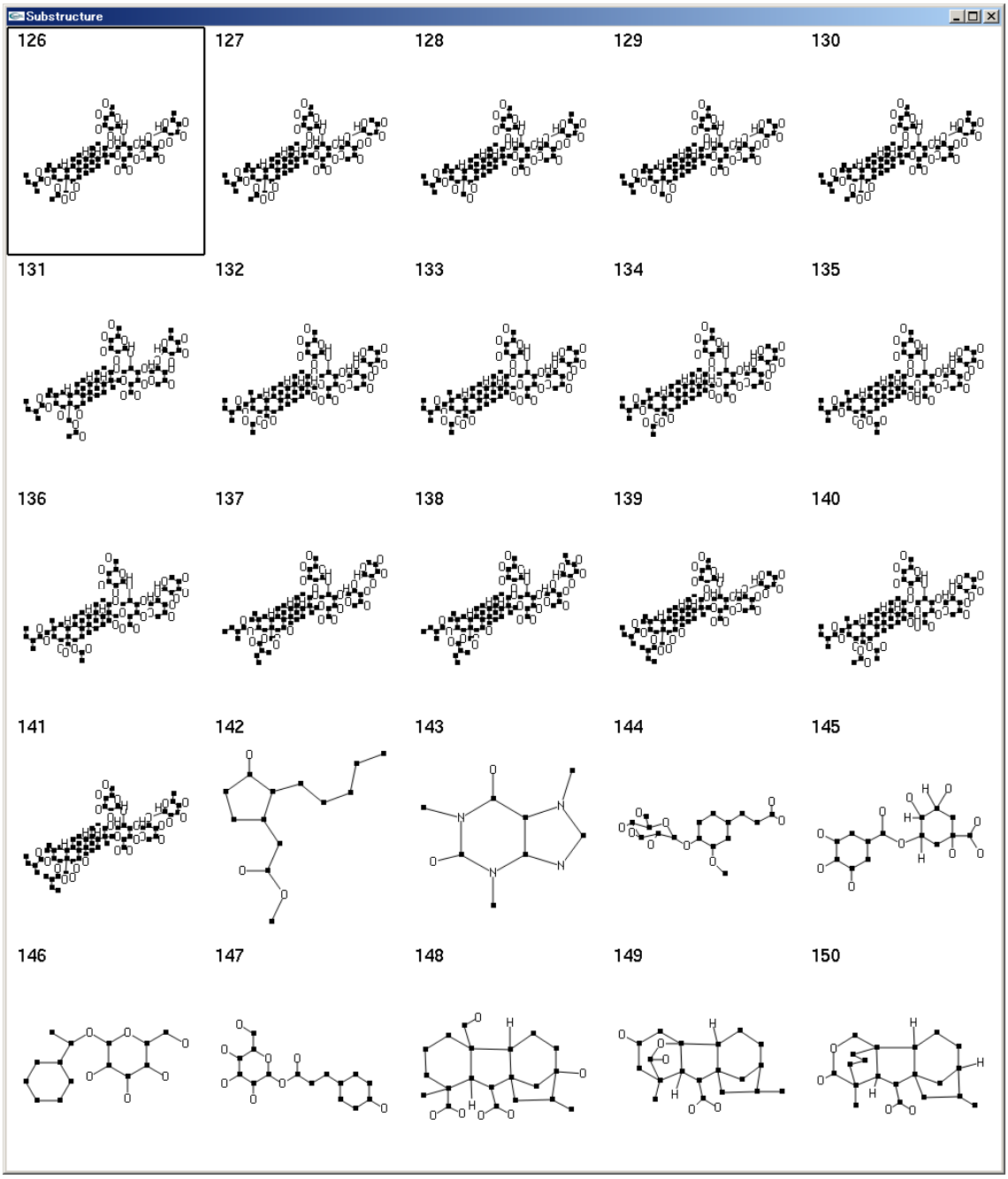


図 89 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 6

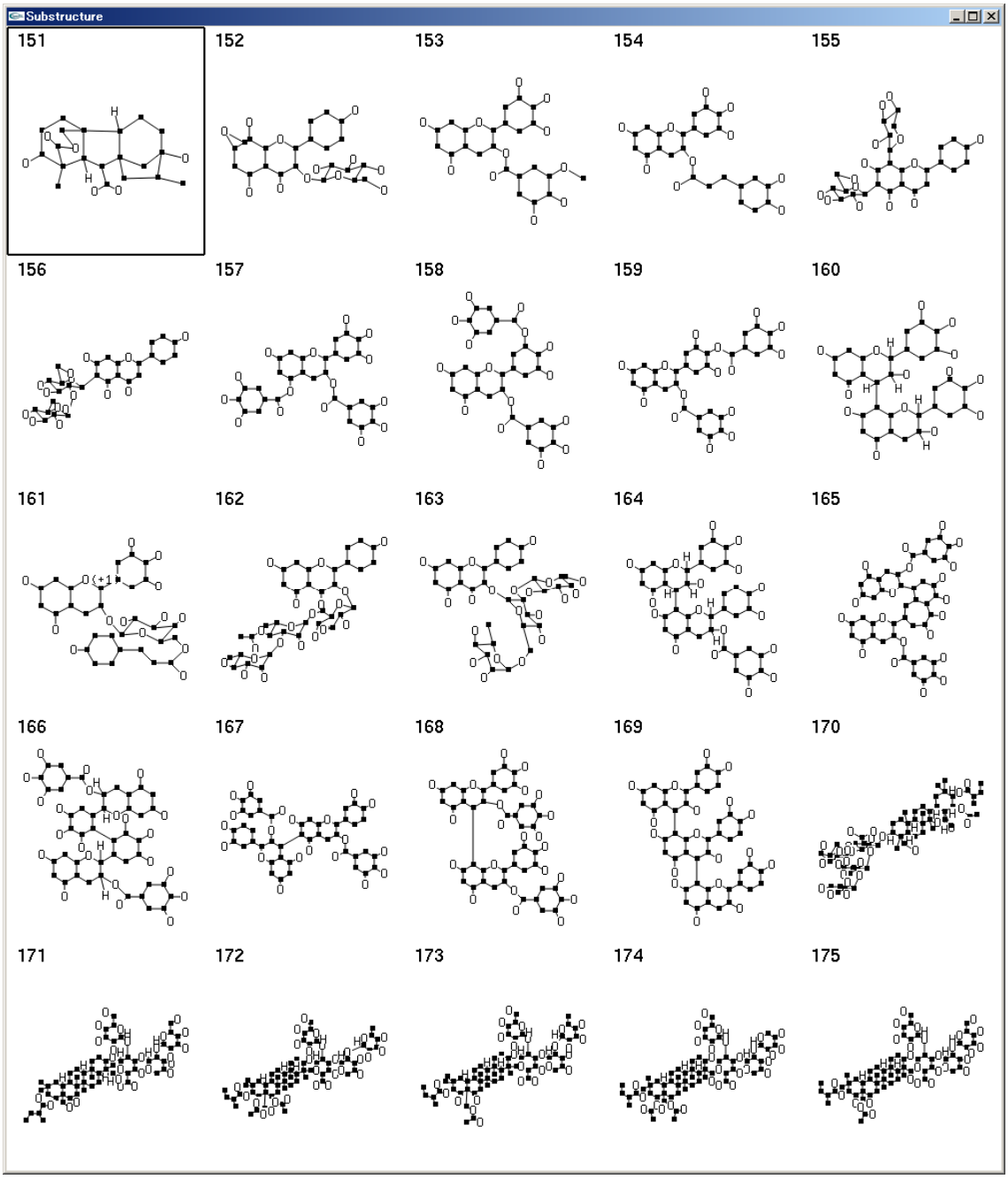


図 90 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 7

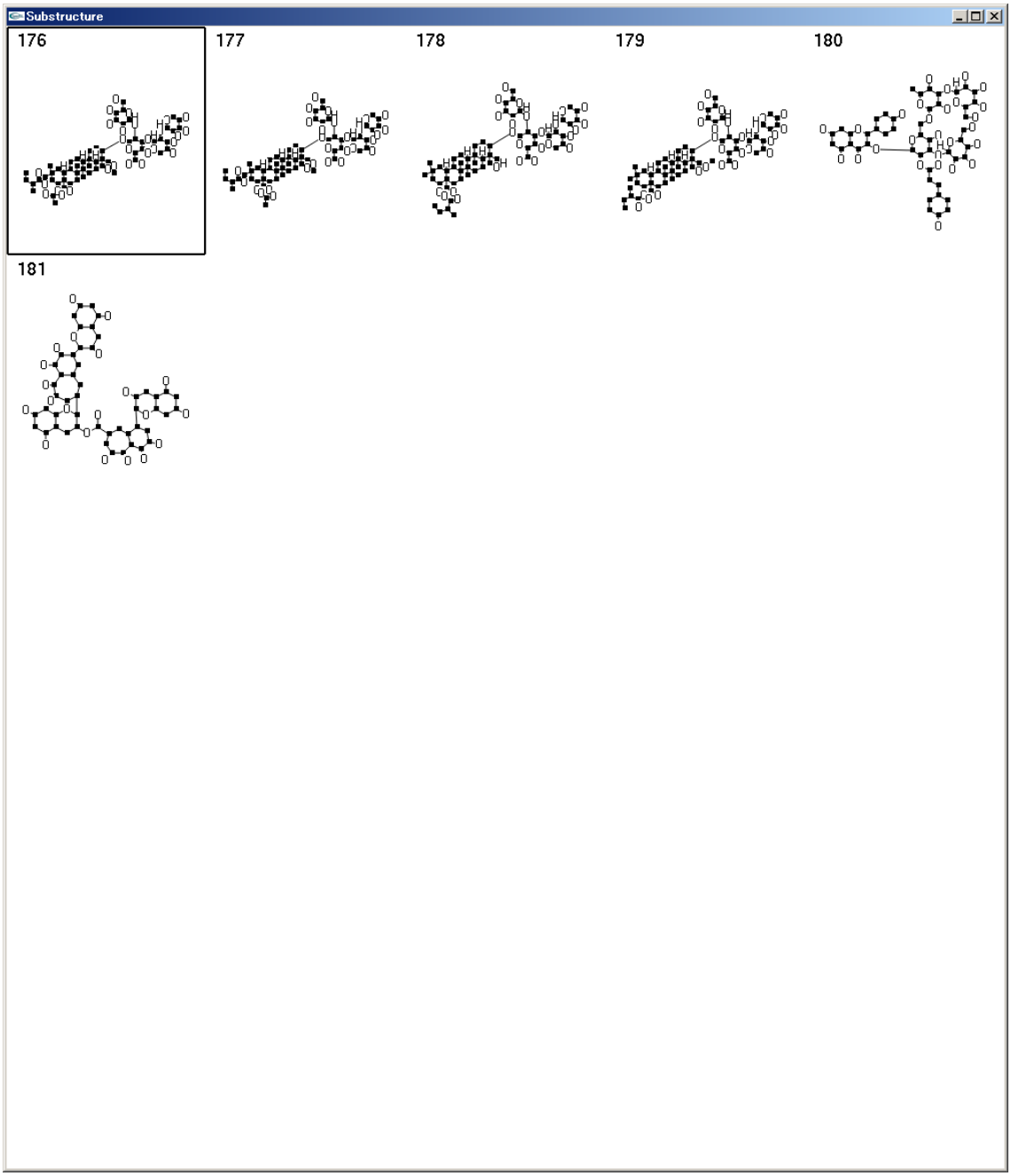


図 91 *Camellia sinensis* が保有する 123 代謝物から抽出した Framework-based MFCS 8

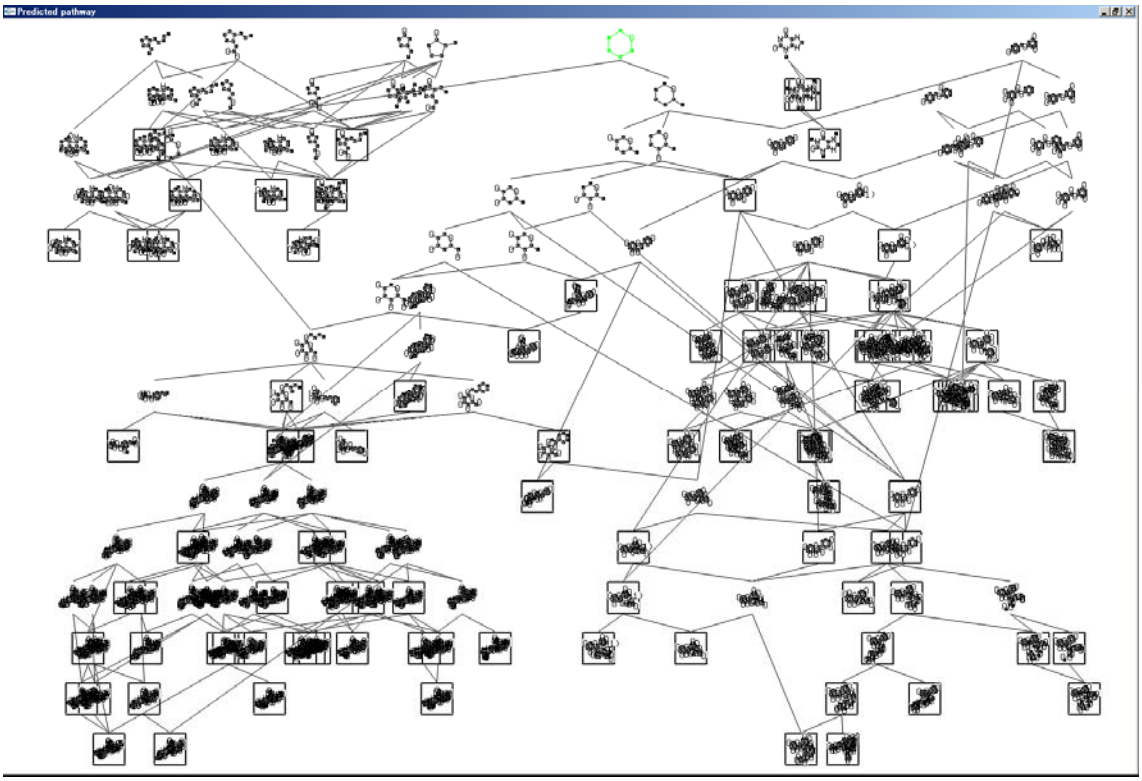


図 92 Framewokr-based MFCS を用いた Compound レベルの代謝経路予測

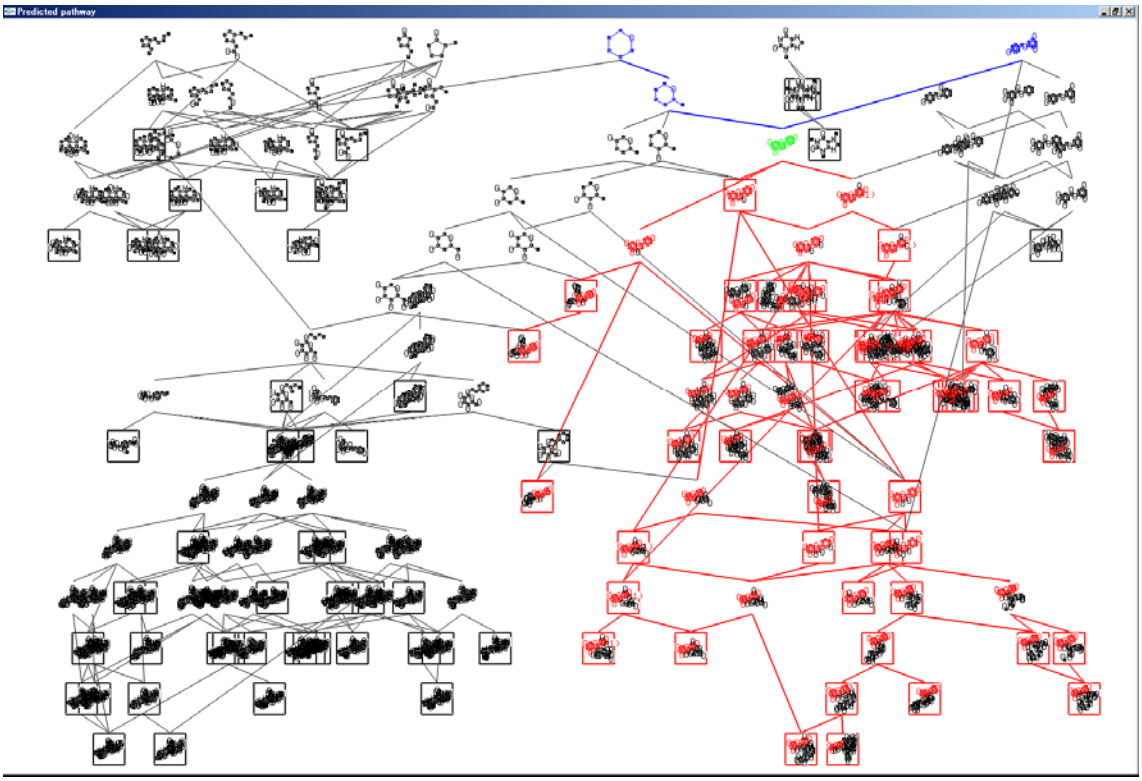


図 93 Compound レベルの代謝経路予測結果(Flavonoid のクラスタ)

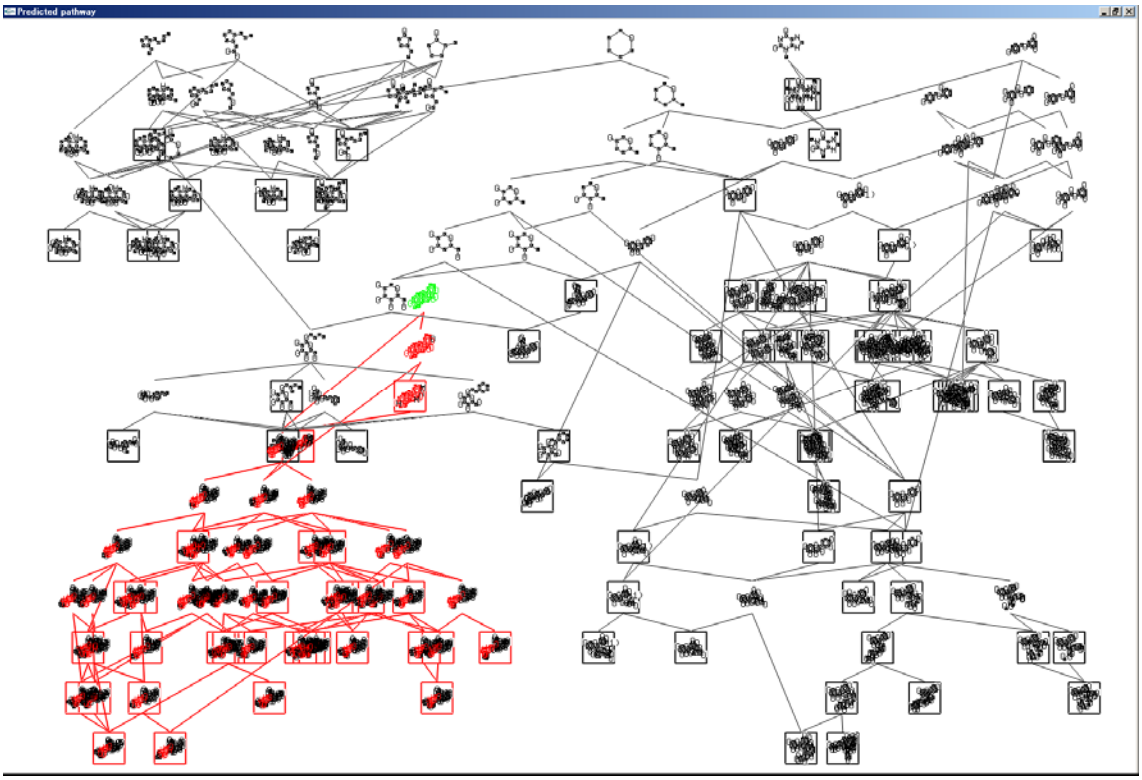


図 94 Compound レベルの代謝経路予測結果(Steroid のクラスタ)

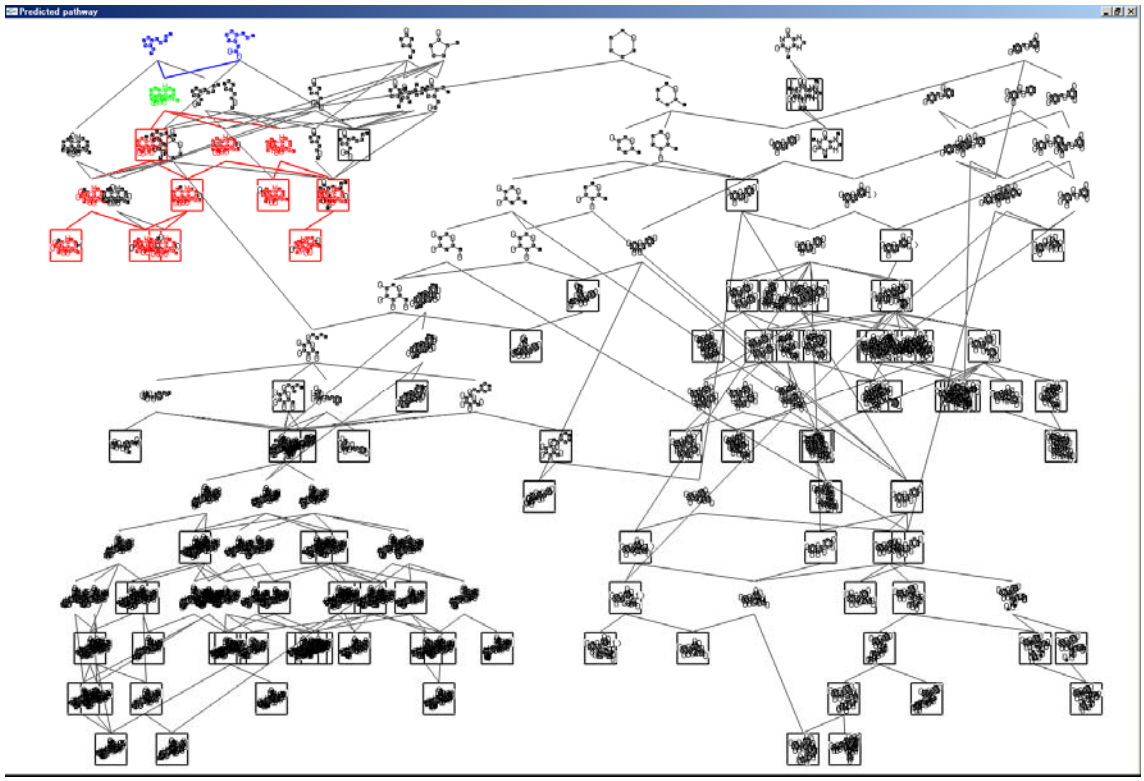


図 95 Compound レベルの代謝経路予測結果(Gibberellin のクラスタ)

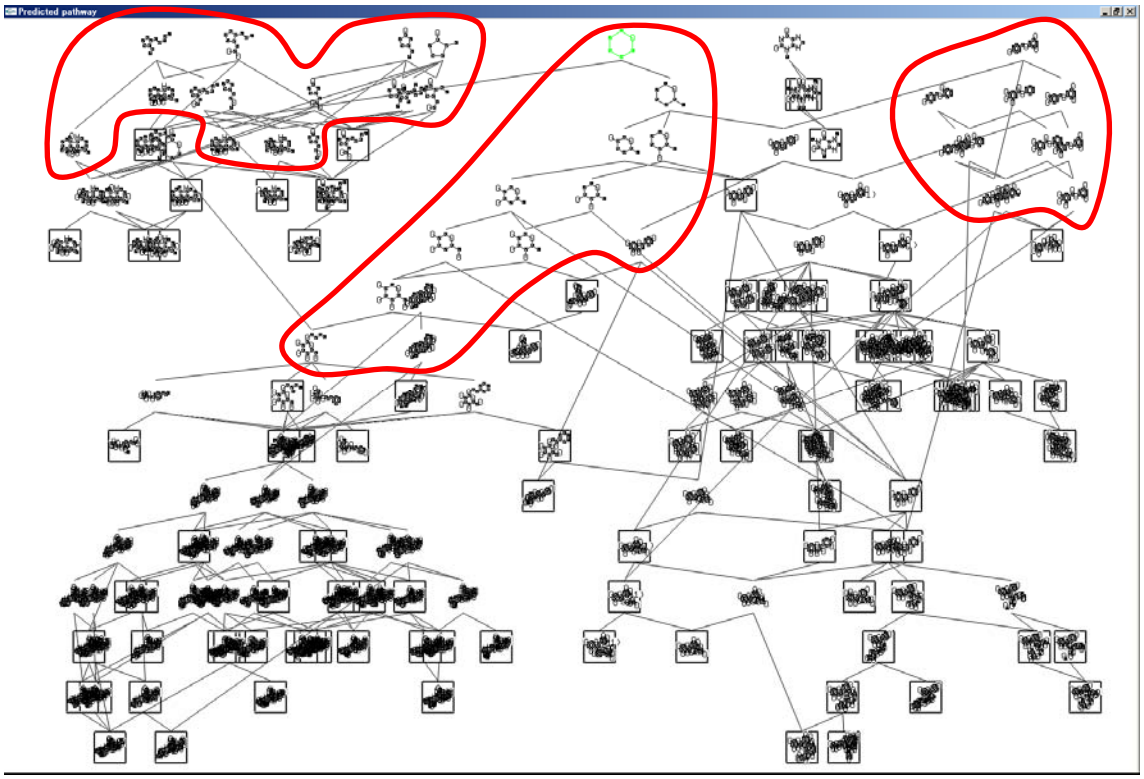


図 96 Compound レベルの代謝経路予測結果(中間 Subgraph のクラスタ)

第5.2.2項　　KNApSAcK 全データを用いた代謝経路予測

本項では、KNApSAcK に含まれる 34,653 代謝物データを用い、大規模なデータに対して提案手法が動作可能かどうかの検証を行った。入力とした 34,653 Chemical graph から 690,695 Framework-based MFCS を抽出することが出来た。この内 93%の Chemical graph に対しては、Framework-based MFCS の取得にかかった時間が 1 分未満であった(図 97)。残り 7%の Chemical graph はペプチド結合の様な分岐の数が多い Side chain を保有する Chemical graph であった。これらに対しては、第 4.1.2 項で実装した、制限時間を 1 分間に設定する処理により、探索を途中で打ち切ることで対応した。全ての Chemical graph から Framework-based MFCS の取得にかかった時間は約 56 時間であった。

抽出した 690,695 Framework-based MFCS から Parent-Child の関係にある Framework-based MFCS ペアを抽出することで代謝経路予測を行ったところ 4,360,618 代謝経路を予測した。代謝経路予測を行う際には 7 台のノート PC で分散処理を行い約 3 日で計算が完了した。各 PC に用いた処理時間を合計すると、仮に 1 台のノート PC で全ての処理を行った場合、約 2 週間で代謝経路予測処理が完了する結果となった。2 週間は計算コストとしては大きいと感じるかもしれないが、先行研究において 15,050 代謝物間の経路予測に対して計算が完了出来たのが 3.34%であった点を踏まえると大幅な進歩であると言える。

また、計算時のメモリ使用量は 700MB 程度であった。今後 KNApSAcK の化学構造データが増えたとしても、構造が決定されている代謝物の総数が約 50,000 であるとの報告(Verpoote, 1998; De Luca and St Pierre, 2000)を考慮すると、提案手法を用いることで代謝経路予測を行うことが十分可能である。

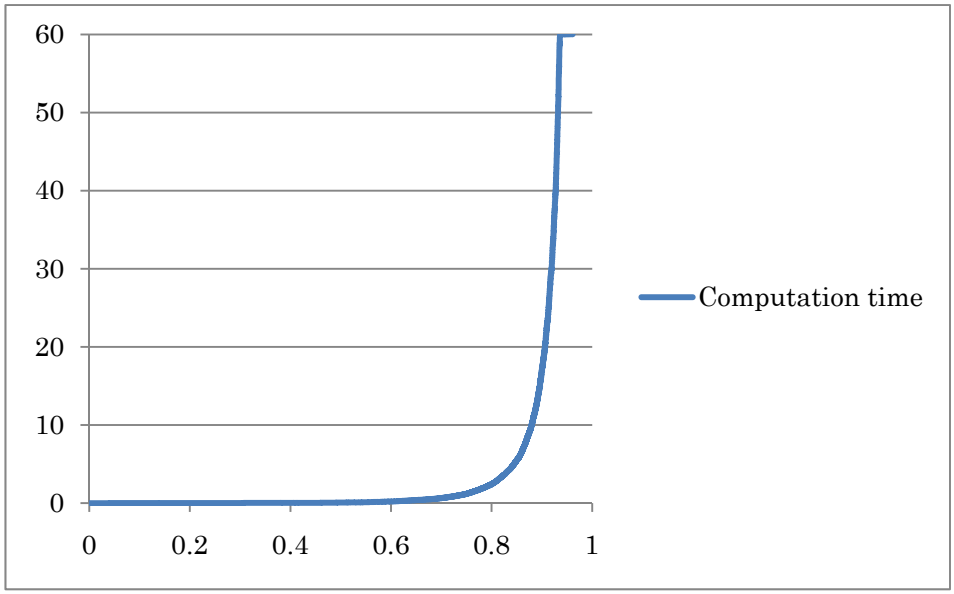


図 97 1 つの Chemical graph に対する Framework-based MFCS の取得時間の累積分布
縦軸は 1 つの Chemical graph に対する Framewrok-based MFCS の取得時間(秒)、横軸は Chemical graph の累積分布を表す。KNApSAcK の 34,653 代謝物の Chemical graph から Framework-based MFCS の抽出を試みた結果、93%の Chemical graph からは 1 分以内に、さらに 90%の Chemical graph からは 10 秒以内に Framewrok-based MFCS の取得が完了した。取得に時間がかかるケースは Chemical graph がペプチド結合の様な分岐の数が多い Side chain を保有する場合であった。

第6章　結論

本研究では、現在大部分が未解明な代謝経路を解明することを目的として、既知の代謝経路上で基質と生成物の Framework 間に包含関係が認められることをヒューリスティックとして導入した Framewrok-based Maximum Frequent Connected subgraph 抽出手法を提案した。Framework を含む Connected subgraph に限定して、Maximum Frequent Connected subgraph の抽出を行うことで、抽出処理の大幅な高速化を実現した。また、化合物ペアが代謝経路であるか否かを判定する為の手法として、graph-based approach による部分グラフの包含関係に着目した代謝経路予測手法を提案した。提案手法を用いることで、既知の代謝経路の大部分を予測できていることが確認でき、新規の代謝経路を予測していることも確認できた。提案手法により KNApSAcK に含まれる全 34,653 代謝物に対して経路予測を行った結果、2 週間で計算処理を完了させることが出来た。先行研究において 15,050 代謝物間の経路予測に対し、計算が完了出来たのが 3.34%であった点を踏まえると大幅な進歩である。

第II部　代謝経路可視化

第7章　序論

第 I 部で行った代謝経路予測は、どの代謝物のペアが代謝経路上で隣接関係にあるのかを予測した。予測した代謝経路を理解する為には、各代謝物を最適に配置し可視化する必要がある。本章では第 7.1 節においてグラフ描画に関連する先行研究を紹介した後、第 7.2 節において本研究の目的を明確にする。

第7.1節　関連研究

本節ではグラフ描画に関する先行研究を紹介する。第 7.1.1 項では、グラフ描画アルゴリズムの先行研究の中でも非常に美しい結果の得られる LGL について説明する(Adai et al., 2004)。第 7.1.2 項では、システム生物学分野で広く使われている、ネットワークデータ統合・解析・可視化の為の Java ベースのオープンソースプラットフォームである Cytoscape について説明する(Shannon et al., 2003; Cline et al., 2007)。

第7.1.1項　バネ力を用いたネットワーク描画

グラフ描画のアルゴリズムとして最も多い試みは、グラフのエッジにバネ力を設定し、ノード間に反発力を設定することでグラフを描画するアプローチである。これらのアプローチの中でも Adai らが提案している LGL(Adai et al., 2004)は、描画結果が非常に美しく、各ノード間の関係を認識しやすい。しかし、バネ力を用いた手法では多くのデッドスペースが発生してしまう(図 98)。

本研究では、代謝経路を予測した後、化学構造を代謝マップ上に直接配置することで、構造の変化を視覚的に追えるようなシステムを目指している(第 I 部第 1.2.1 項図 2)。そのためには、与えられた描画空間全体をうまく使ってデッドスペースがなるべくなくなるようなアルゴリズムが求められる。

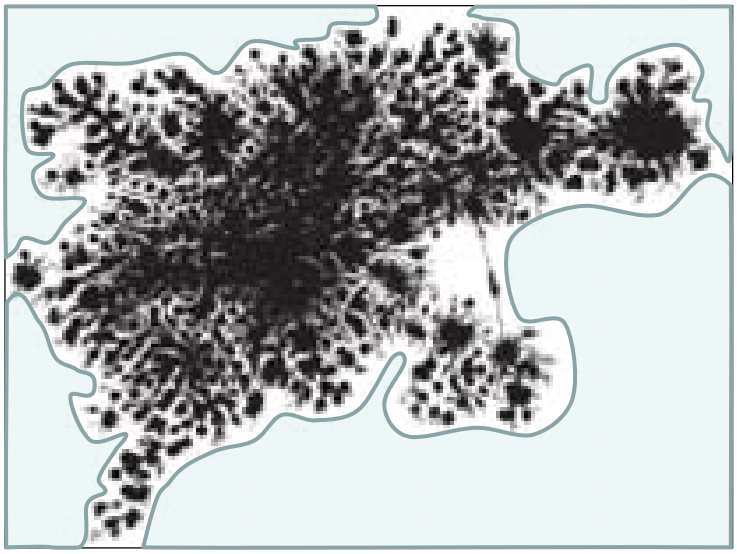


図 98 LGL によって描画されたグラフ(Adai et al., 2004)
LGL に代表されるバネ力を用いたアルゴリズムは、非常に美しい出力結果が得られ、ノード間の関係も認識しやすい。一方でデッドスペースが存在してしまう。

第7.1.2項　　Cytoscape

Cytoscape は UC, San Diego のグループによって運営されている、システム生物学分野で広く使われている、ネットワークデータ統合・解析・可視化の為の Java ベースのオープンソースプラットフォームである(Shannon et al., 2003; Cline et al., 2007)。数多くのアルゴリズムが Cytoscape のプラグインとして提案されている。ネットワーク解析に関するアルゴリズム開発を研究者が考えた場合、それをソフトウェアとして提供する為には、画面制御、ファイルの入出力、ユーザインターフェースの構築等、本質的なアルゴリズム以外の部分に関する多くの開発を行う必要がある。Cytoscape の利点は、これらの共通機能をプラットフォームとして提供することで、開発者はアルゴリズムの開発のみに専念できる点であり、ユーザは複数の研究者によって開発されたシステムを一つのシステム上で利用することが可能になる点である。

しかし、この点はデメリットにもなる。例えば大規模なネットワークを扱おうとした際、巨大なメモリ空間が要求される様なケースがしばしば存在する。対策としてデータ構造レベルで特殊な対応を行うことが必要になるケースが存在する。しかし、データ構造を変化させるような対応は、そのデータ構造を利用しているあらゆる機能に影響を与える。システムに汎用性を求めている場合、この様なケースに対し柔軟な対応が出来なくなる。実際、Cytoscape も大規模アップデートを行う際に、「以前のバージョンの Cytoscape で動作していたソフトウェアのいくつかは動作しなくなる。」とアナウンスをした上でバージョンアップを行っている。このような議論は、個々の機能の性能を優先させるか、汎用性を備えた

システムを構築するか、といった設計思想の違いによるものである為、利点と欠点が出るのは当然のことである。

Cytoscape のメーリングリスト内の議論になるが、2009 年 10 月あるユーザが「9,000 個の蛋白質から構成されるネットワークの描画に失敗する。」といった質問を投げかけたことがあった。それに対する開発者の一回目の返答は「巨大なネットワークを扱おうとした場合、パフォーマンスはハードウェアの性能に依存する。私は 2.53GHz の CPU と 4GB のメモリからなるマシンを持っているが、BINDhuman.sif の 19,000 ノードと 31,000 エッジからなるネットワークの描画をしようとは思わない」という内容であった。一方で、本研究で開発したシステムは、690,695 ノードと 4,360,618 エッジからなるネットワークを処理するのに 700MB 程度のメモリで動作可能である（使用したマシンは 2.33GHz の CPU と 1GB のメモリで構成される）。第 I 部第 4.2 節で述べたが、当初、690,695 Subgraph 間の包含関係を全て計算するようなアルゴリズムにしていた為、690,695×690,695＝477,059,583,025 のデータ量が必要となり少なく見積もっても約 480GB のメモリが必要なシステムであった。しかし、問題箇所を特定し、データ構造を見直し、関連するアルゴリズムを全て修正することで、2 日程度で問題解決が出来た。この様な対応が可能なのは、全てのシステムを自前で構築している為、全体のパフォーマンスを落とすことなく、大容量データを扱うことが可能かどうかを即座に検討できる点にある。第 I 部第 1.1.1 項で述べたように、約 50,000 の代謝物の構造が決定されていると言われており(Verpoorte, 1998)、自然界には約 200,000 の代謝物が存在すると言われている(Hostettman and Terreaux, 2000)。この様な大規模なデータが本研究の対象であり、汎用性を重視している Cytoscape では、先のメーリングリストにおけるやり取りを見る限りこの様な大規模データを取り扱うことはできないと考えられる。

また、近年では GPU を汎用計算に用いる GPGPU の様な概念や、CPU のマルチコア化等、アーキテクチャの変化を伴ったハードウェアの進歩が見受けられる。この様な状況でハードウェアの進歩に伴う高速化の恩恵にあずかろうとした場合、ハードウェアのアーキテクチャを理解してシステムを構築する必要がある。プラットフォームはハードウェアに近い位置に存在する処理である為、これらの影響を最も受ける部分である。開発効率を重視して、プラットフォームの開発を他者に任せる選択は、この様な最新ハードウェアのアーキテクチャに触れる機会を失うことにも繋がる。ユーザの視点に立って、複数の開発者が開発した機能を 1 つのシステム上で利用できるというのはとても大きな利点ではあるが、開発コストの低減を目的として外部ライブラリを利用するという選択は、長期的な視点で考えた場合ノウハウを蓄積することが出来なくなることにも繋がり、他者との勝負に勝てなくなることを意味するのではと考えている。

本研究では、AntTweakBar、OpenGL、GLUT 等の GUI ライブラリは用いているが、アルゴリズムの性能に直接影響が発生するようなライブラリは用いていない。

第7.2節　　研究目的

本研究では、第 I 部で予測した代謝経路を描画する際、描画領域に化学構造を直接配置することで、構造の変化を代謝経路全体にわたって認識することを目的とする。そこで、PC のディスプレイや論文の紙面等、与えられた描画領域をデッドスペースなく有効に利用して、各代謝物を最適に配置するアルゴリズムの開発を本研究の目的とする。

第8章　提案手法

本研究では、非線形多次元データの可視化に用いられる「自己組織化マップ」(Kohonen, 1982; Kanaya et al. 2001)のアルゴリズムに着目し、自己組織化にかかわるアルゴリズムを代謝経路可視化アルゴリズムに応用した。本章では、まず第 8.1 節において自己組織化マップの動作原理を説明する。次に第 8.2 節において、提案手法で自己組織化マップのアルゴリズムを代謝経路可視化アルゴリズムにどのように応用したのかについて説明を行う。最後に第 8.3 節において予測した代謝経路の包含関係を表現することを目的とし、包含関係に基づいて各 Subgraph を階層表示する方法について述べる。

第8.1節　　自己組織化マップの応用

自己組織化マップでは、多次元空間に散らばるデータ群に格子状グラフを張りつかせた後、格子状グラフの座標系に多次元データを写像し、格子状グラフの二次元座標系で多次元データを表現している。自己組織化マップの動作原理は以下の通りである。

(1) 収束するまで操作(2)～(5)を繰り返す。
  (2) 多次元データをランダムに一つ選択する(図 100)。
  (3) 選択したデータと最も近い格子状グラフのノードを探索する(図 101)。
  (4) 最近隣ノードに隣接しているノードを探索する(図 102)。
  (5) 最近隣ノードと隣接ノードを選択したデータに向かって移動させる(図 103)。
(6) 格子状グラフが多次元データにフィットする(図 104)。
(7) 格子状グラフの座標系に多次元データを写像し、格子状グラフの座標系で多次元データを可視化する(図 105)。

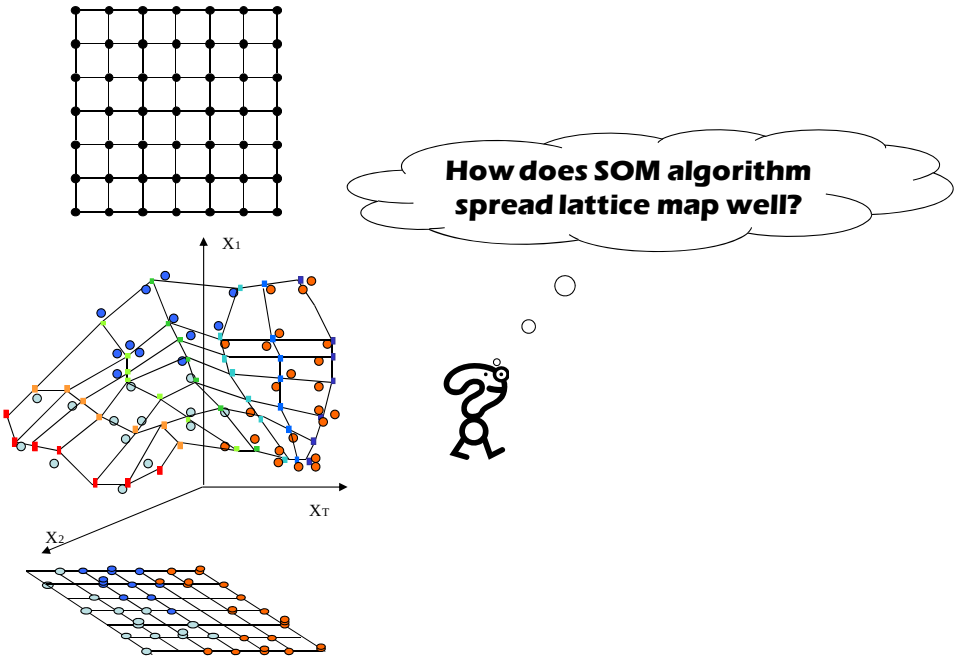


図 99 自己組織化マップのイメージ

本研究では格子状グラフが多次元データにフィッティングする仕組みを代謝マップの最適配置アルゴリズムに応用した。

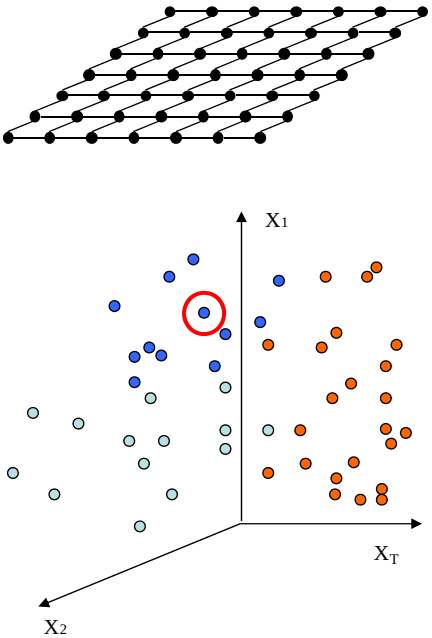


図 100 ランダムにデータを選択

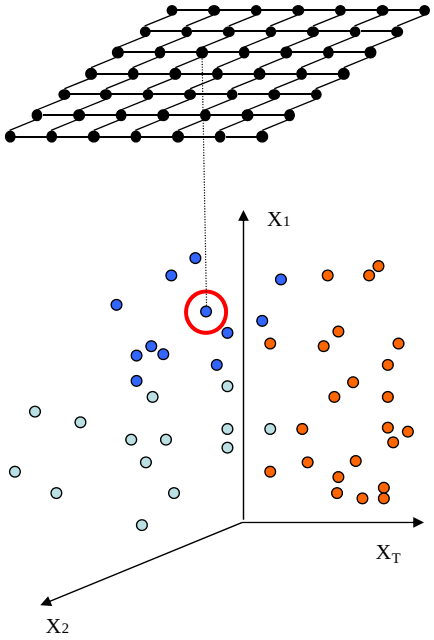


図 101 選択したデータと最も近い位置にある、格子状グラフのノードを探索

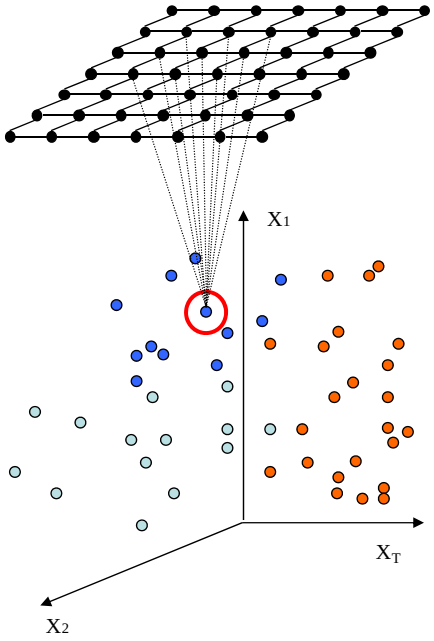


図 102 最近隣ノードに隣接しているノードを探索

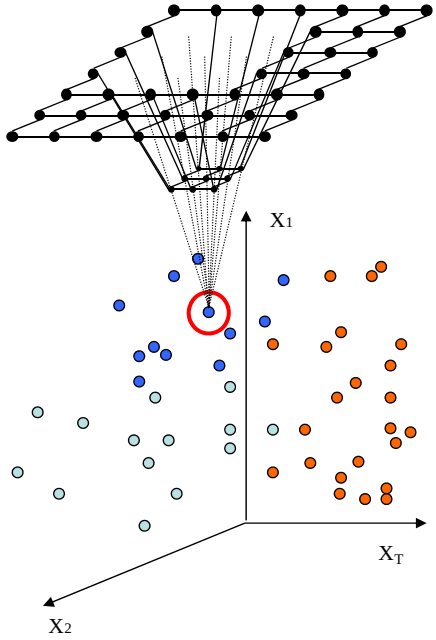


図 103 最近隣ノードおよび、隣接ノードを選択したデータに向かって移動

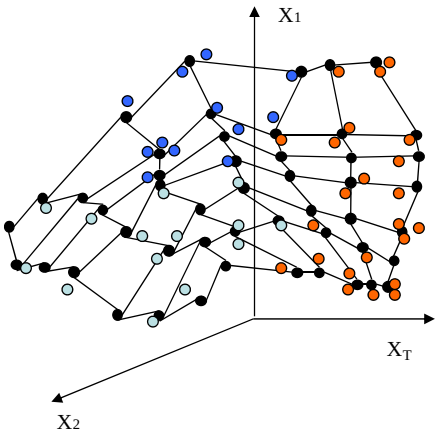


図 104 多次元データにフィッティングした格子状グラフ

収束するまで図 100～図 103 の操作を繰り返すことで、格子状グラフが多次元データにフィットする。

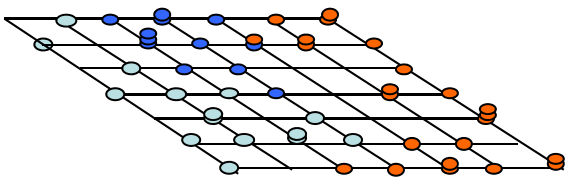


図 105 自己組織化マップによる多次元データの可視化結果

格子状グラフの座標系に多次元データを写像することで可視化する。

第8.2節　　Network Self-Organization

提案手法は、自己組織化マップの動作原理について、「格子状グラフ」を「予測した代謝経路」、「多次元データ群」を「代謝経路を展開させたい 2 次元平面（もしくは 3 次元空間）上のランダムな座標」、「隣接ノード」には「最近隣ノードの Ancestor subgraph および Descendant subgraph」をそれぞれ対応させることで、グラフの最適配置アルゴリズムとして応用する。

自己組織化マップの動作

(1) 与えられたデータの中からランダムに 1 データを選択
(2) 格子状グラフの全ノードから最近隣ノードを探索
(3) 最近隣ノードと隣接しているノードを探索
(4) 最近隣ノードおよび隣接ノードを選択したデータに向かって移動

Network Self-Organization の動作

(1) 代謝経路を展開させたい空間からランダムに座標を取得
(2) 取得した座標に最も近い位置に存在する最近隣 Subgraph を探索
(3) Subgraph の Ancestor subgraph および Descendant subgraph を探索
(4) Subgraph、Ancestor subgraph、Descendant subgraph を選択座標に向かって移動

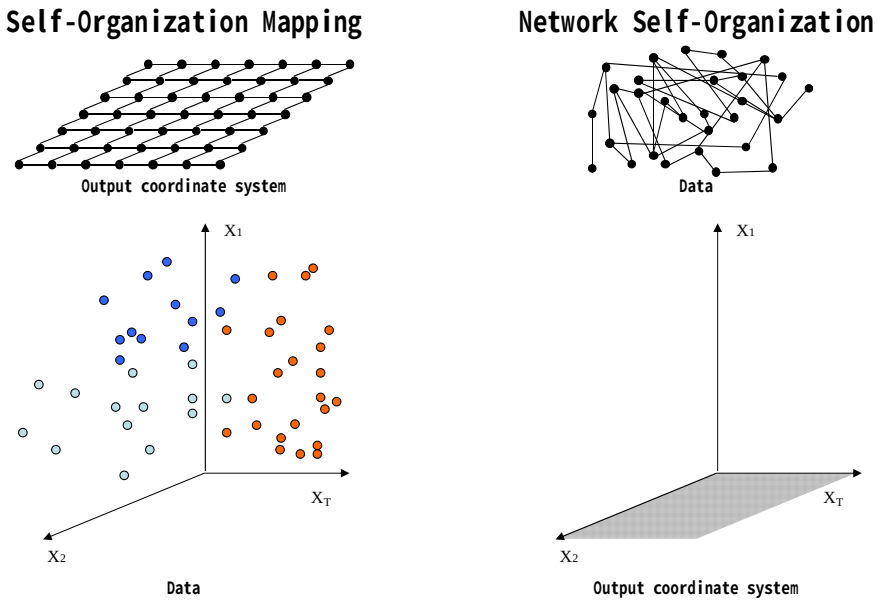


図 106 自己組織化マップと提案手法の対比
自己組織化マップでは、格子状グラフを多次元データにフィッティングさせる。
提案手法では、予測した代謝経路を展開したい領域全体にフィッティングさせる。

ここで、Subgraph の移動量を決める際の条件として以下を設定した。

- 最近隣 Subgraph の移動量(A)は、「Network Self-Organization の動作(1)」で取得した座標との距離(B)を最大とする。
- A の大きさは Gain で調整が可能である。調整後の移動量を B(=A×Gain)とする。
- Ancestor(or Descendant) subgraph の移動量は最近隣 Subgraph との最短経路長 Length によって決める。ここで、Length=0 は最近隣 Subgraph 自身を指す。
- 最短経路長(Length)を変数として、Length の小さな Subgraph はより追従させ、大きな Subgraph は追従しないようにする。
- Length=0 の時、移動量は B とする。
- Length→∞の時、移動量はマイナスの値を取らない。(Length→∞において移動量は 0 に収束する)
- 減衰係数(Attenuation)によって、追従量を調整できる。
- Attenuation を大きくした場合、近くの Subgraph のみを追従させたい。

以上の目的を満たす Subgraph の移動量を計算する式として、以下の式を採用した。

$$\mathrm{After} = \mathrm{Before} + (\mathrm{Target} - \mathrm{Before}) \times \mathrm{Gain} \times (1 + \mathrm{Attenuation})^{-\mathrm{Length}^2}$$

各変数は以下に示す値を表している。

After：移動後の座標
Before：移動前の座標
Target：ランダムに選んだ座標
Gain：利得($0 \leq \mathrm{Gain} \leq 1$)
Attenuation：減衰係数($0 \leq \mathrm{Attenuation}$)
Length：最近隣 Subgraph からの最短経路長

減衰係数を調整することで、Ancestor subgraph および Descendant subgraph の移動量は Subgraph との最短経路長によって図 107 の様に変化する。減衰係数が 0 の場合、Ancestor subgraph および Descendant subgraph は Subgraph に完全に追従する。減衰係数を上げていくと、Subgraph から離れている要素の移動量は徐々に減少していく。実装したソフトウェア上では Gain と Attenuation を調整できるようにし、手動による Simulated Annealing（焼きなまし法）を実装した。自己組織化マップにおいて、自己組織化を担当しているアルゴリズムをネットワークの最適配置アルゴリズムとして応用していることから、本アルゴリズムを「Network Self-Organization」と名付けた。

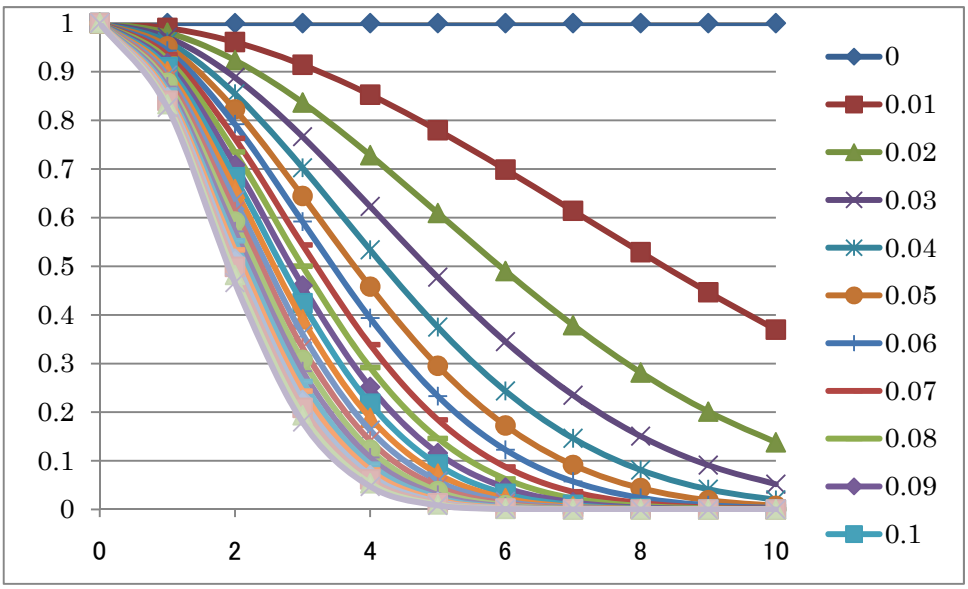


図 107 減衰係数を変化させた際の移動量の変化

横軸は最近隣 Subgraph と Ancestor(or Descendant) subgraph との経路長

縦軸は移動量である。減衰係数を大きくすることで、最近隣 Subgraph の近くに位置する Subgraph の移動量は大きくなり、遠くに位置する Subgraph の移動量は小さくなる。

第8.3節　　階層表示

第 I 部で提案した包含関係に着目した代謝経路予測手法は、入力とした Subgraph から Parent-Child の関係にある Subgraph のペアを代謝経路として予測している。そこで、代謝経路を描画する際に、全ての Parent-Child ペアにおいて、Parent subgraph が Child subgraph よりも画面上部に配置されるように階層表示することで、包含関係の認識を容易にする機能を取り入れた。各 Subgraph をどの階層に表示するべきかは以下の手順で計算する。(1),(2)は初期化処理である。(3)～(7)の処理によって、Child subgraph の階層が Parent subgraph の階層よりも大きくなるように設定を行っている(図 108)。(8)～(9)の処理により、(3)でルート Subgraph の階層を 1 と固定した結果、距離が離れた Subgraph を Child subgraph に近付けている(図 109)。

(1) Subgraph の数だけ箱を用意する。
(2) 全ての Subgraph の階層情報に 0 をセットする。
(3) Parent subgraph を持たないルートとなる Subgraph を箱の先頭にセットする。この時セットした Subgraph の階層を 1 とする。
(4) 箱の先頭の Subgraph から順に処理(5)～(7)を実行する。
  (5) Child subgraph が箱に追加されていない場合、箱に追加する。
  (6) 「Child subgraph の階層情報 ＜Subgraph の階層+1」の場合、Child subgraph の階層情報 ＝Subgraph の階層+1 とする。
  (7) 次の Subgraph の処理へ移る。
(8) 箱の末端の Subgraph から順に(9)～(11)の処理を実行する。
  (9) Child subgraph が存在しない場合は次の Subgraph の処理へ移る。
  (10) 全ての Child subgraph から最小階層情報(MinLayer)を取得する。
  (11) Subgraph の階層情報を MinLayer-1 にする。

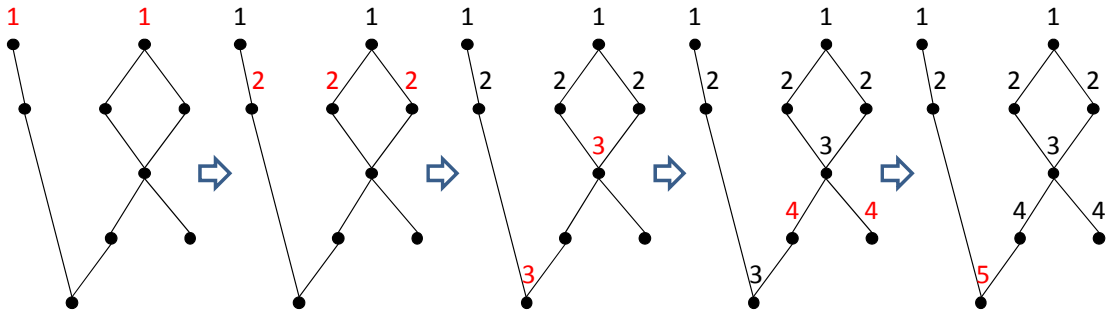


図 108 階層情報の設定 1

Parent subgraph を持たない Subgraph の階層を 1 に設定する。その後、階層の設定が済んだ Subgraph A の Child subgraph に対し、「Subgraph A の階層+1」を階層として設定する操作を繰り返すことで、全ての Subgraph に階層を設定する。

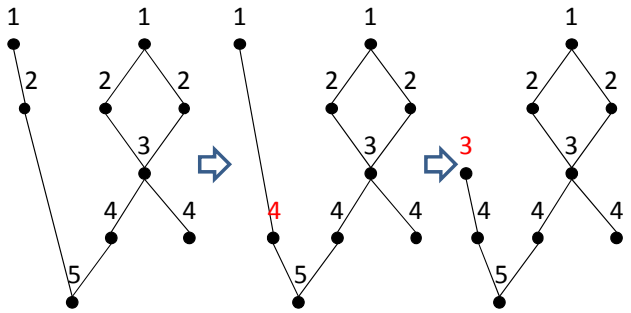


図 109 階層情報の設定２

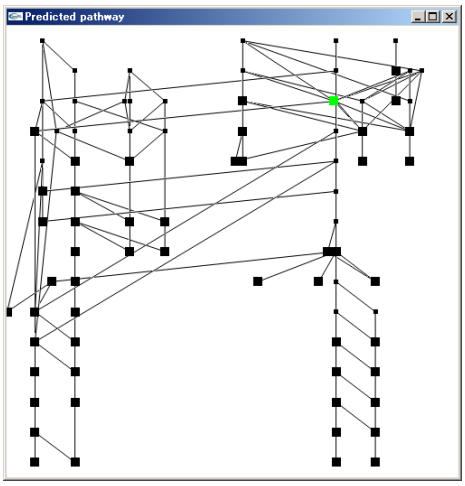
Child subgraph と離れた Parent subgraph の階層を Child subgraph に近付ける

第9章　結果と考察

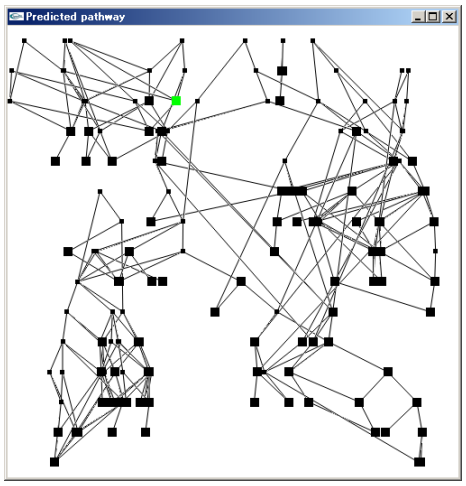
本研究で開発したシステムは、第 III 部第 11 章で述べる MetClassifier に実装されている。MetClassifier では、代謝経路を描画する際のモードとして、描画領域の選択として 2 次元もしくは 3 次元、描画方法の選択として、Randam、Hierarchy、Spread が選択できる。Randam を選択した場合、全ての Subgraph が選択した描画領域内にランダムに配置される。Hierarchy を選択した場合、垂直方向は階層情報で固定し、水平方向と奥行き方向は Network Self-Organization によって座標を決定する。Spread を選択した場合は、全ての方向に対して Network Self-Organization によって座標を決定する。

ここでは、第 I 部第 5.2.1 項で用いた、*Camellia sinensis* の代謝経路予測結果を用いて、2 次元の Hierarchy 描画を例に、Gain や Attenuation を変化させることで、描画結果がどのように変化するのか確認する(図 110)。図 110 は Subgraph をドットで表示している。Gain を上げると代謝経路全体がばらつき、Gain を下げると収束する。また、Attenuation が 0 の場合は、Ancestore-Descendant の関係にある Subgraph 群が同調して動くのに対し、Attenuation を上げることで予測代謝経路上において、近くに位置する Subgraph のみが追従し、遠くに位置する Subgraph は離れていくことが分かる。

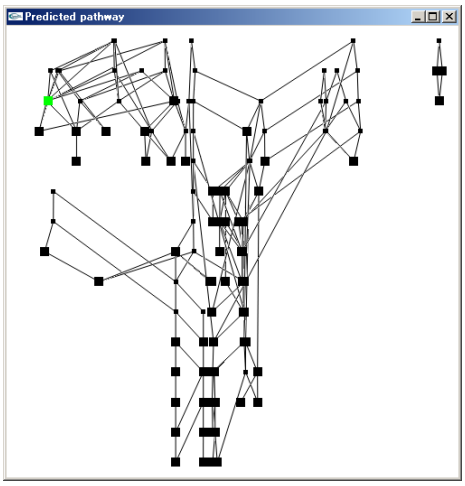
描画画面を見ながら、代謝経路の展開結果が局所解に落ちていると判断出来た場合は、Gain を上げるもしくは画面上の node(Subgraph)を直接マウスでドラッグする等して局所解から脱出させた後に、Gain を小さくすることで、最適解に近付けることが出来る。Attenuation の値を小さくすると、大域的にどのような代謝経路が存在するかを理解することが出来る。Attenuation の値を大きくすると、各 Subgraph が描画領域全体に散らばるように動作する。その結果、一つの Subgraph に割り当て可能な領域が広がり、各 Subgraph をドット表示ではなく Subgraph で表示する為の領域を得ることが出来る。これにより、予測代謝経路上における化学構造の変化を把握することが可能となる(図 111)。



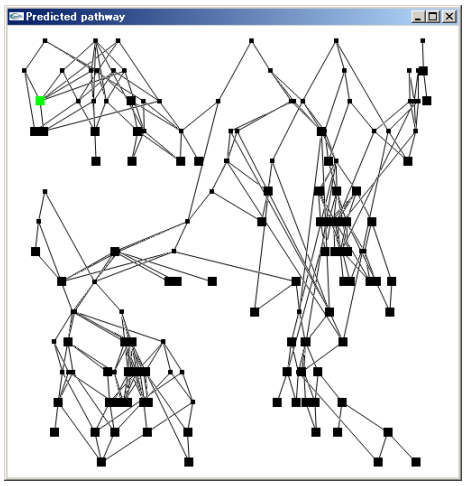
Gain=1 Attenuation=0



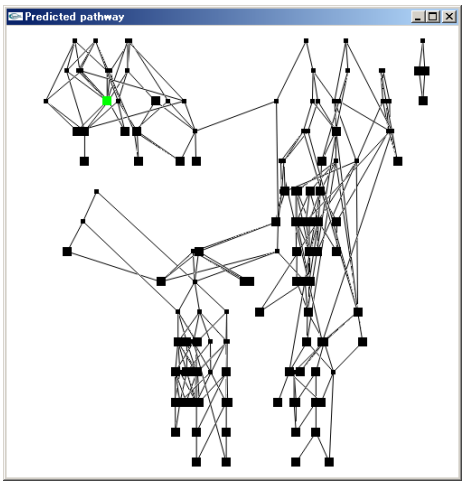
Gain=1 Attenuation=0.1



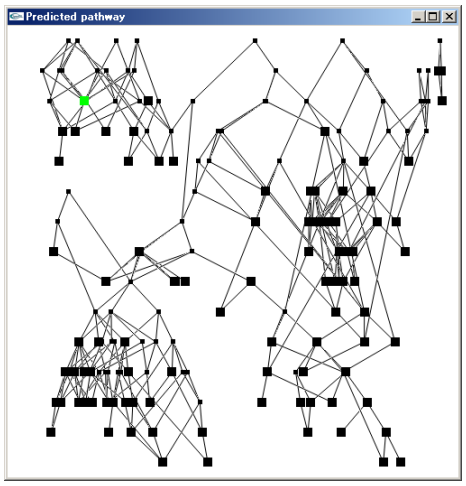
Gain=0.5 Attenuation=0



Gain=0.5 Attenuation=0.1



Gain=0.1 Attenuation=0



Gain=0.1 Attenuation=0.1

図 110 Gain と Attenuation の変化と対応する描画結果

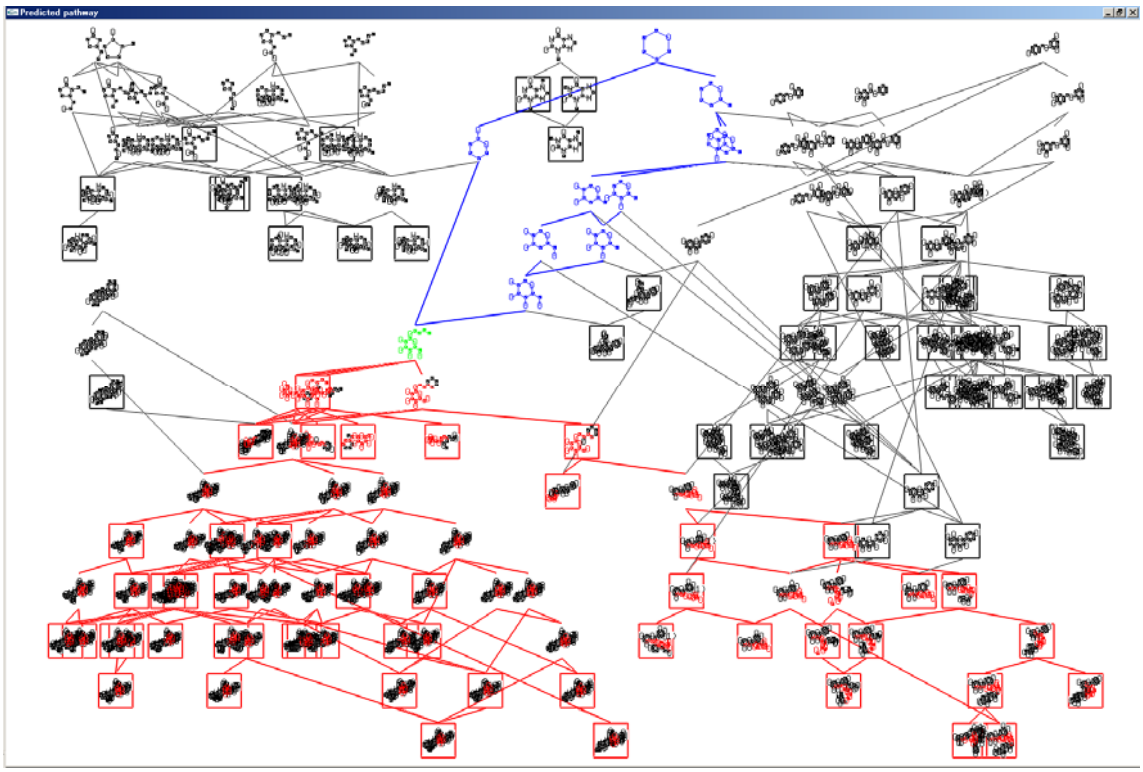


図 111 予測代謝経路の Subgraph 表示

第10章 結論

本研究では、代謝経路の可視化を行う際に、代謝経路上に化学構造を直接配置することで、構造の変化を代謝経路全体にわたって認識することを目的とした。そこで、自己組織化マップのアルゴリズムを応用することで、指定した描画領域に均等に代謝経路を展開するアルゴリズムを提案した。提案手法を用いて代謝経路描画を実行した場合、指定した描画領域全体に均一に各 Subgraph が配置され、一つの Subgraph に割り当てられる領域が広がり、化学構造を把握することが出来る。これにより、ノート PC の画面や論文の紙面のような限られた空間を有効に利用した描画結果を得ることが出来る。

第III部　ソフトウェアの公開

第11章 MetClassifier

第 I 部および第 II 部で説明したアルゴリズムは、MetClassifier として以下のページで公開を行っている(http://kanaya.naist.jp/MetClassifier)。膨大な化合物に対して、いろいろな視点からの解析が考えられる。例えば、「代謝経路が予測したい」とか「既知の化合物と構造が類似している化合物を発見したい」等である。多くのケースでは以下の流れで解析が行われる。

1. データの絞り込み(特定の生物種が保有する、特定の Subgraph を含む etc.)
2. 絞り込んだデータを解析
3. 処理結果の視覚化
4. 処理結果の保存

解析内容に応じて、データの絞り込み方法や解析方法が変化する。各処理に特化したシステムを別途構築した場合、同一の処理を記述したソースコードが複数存在することになる。これは、特定の処理の機能改善を行おうとした場合に、複数存在する全てのソースコードに対して修正をかける必要が出てくることを意味しており、開発効率やメンテナンスを考慮に入れると好ましい状況とは言えない。

MetClassifier では、開発の過程ではオブジェクト指向に基づいて機能を細分化し、類似機能ごとにまとめ、解析時には細分化された機能を組み合わせて利用することで、幅広い解析に対応が可能となっている。第 11.1 節では機能説明を行い、第 11.2 節では具体例を用いて使用方法の説明を行う。MetClassifier の開発にあたり、描画ライブラリとして OpenGL、ユーザインターフェースのライブラリとして GLUT および AntTweakBar、開発環境には Microsoft Visual C++ 2008 Express Edition を利用させて頂いた。

第11.1節　機能説明

MetClassifier には以下の機能が含まれている。

(a) 化合物データベースを開く

- MetClassifier 形式のデータベースを開く
- Molfile 形式の化学構造情報ファイルが保存されているディレクトリを指定する。
- SDfile 形式のファイルを指定する。

(b) アノテーション情報の取り込み

- KNApSAcK の生物種-代謝物関連データファイルの読み込み

(c) 化合物データベースの絞り込み

- 部分構造検索

- Key word による絞り込み「生物種名、組成式、代謝物名、Compound ID が対象」
- リストによる絞り込み
  - Compound ID リスト
  - 生物種名リスト

(d) 絞り込んだ化学構造データから Subgraph を抽出する。

- 抽出済み Subgraph の Import
- Framework
- Maximum Frequent Connected subgraph
- Framework-based MFCS

(e) 化合物データベースの並び替え

- Compound ID
- Subgraph ID[9]
- 原子数[10]
- 結合数[11]
- 部分構造検索実行時にヒットした化合物を自動で TOP に持ってくるかどうかの ON/OFF

(f) 抽出した Subgraph に対して、包含関係に基づいた代謝経路予測を行う。

- 予測済みの代謝経路情報の Import
- 包含関係に着目した代謝経路予測の実行

(g) 予測した代謝経路を可視化する。

- 描画領域の設定
  - 2D
  - 3D
- 描画方法の設定
  - Hierarchical
  - Spread
  - Random
- 同形情報の計算
- Pathway ウィンドウ上における Subgraph 表示方法の切り替え
  - Dot で表示
  - Subgraph ID を表示
  - Subgraph を表示

(h) 各種データの出力

[9] Subgraph として、Framework を取得した場合に利用可能。
[10] Molfile に含まれる原子が対象（非表示の水素はカウントされない）
[11] Molfile に含まれる結合が対象（非表示の水素に関連する結合はカウントされない）

- MetClassifier 形式のデータベース
- Compound ID リスト
- 抽出した Subgraph
- 同形情報(Compound ID と Subgraph ID の二項関係)
- 予測代謝経路情報(Subgraph ID- Subgraph ID の二項関係)
- 可視化結果：各 Subgraph の座標(Subgraph ID-x 座標-y 座標-z 座標)
- Shortest pathway
  (Compound ID－Compound ID の二項関係および中間 Subgraph ID)
- Direct pathway
  (Compound ID－Compound ID の二項関係および中間 Subgraph ID)

第11.2節　　使用例

第 I 部第 5.2.1 項で行った *Camellia sinensis*（お茶）に関する解析を例に説明を行う。

第11.2.1項　　ダウンロードおよび起動

MetClassifier は以下の URL から取得できる(http://kanaya.naist.jp/MetClassifier/)。「1.1 Download MetClassifier」の項目から圧縮ファイルをダウンロード（図 112)。圧縮ファイル解凍後、MetClassifier.exe をダブルクリックすることで起動する(図 113図 114)。

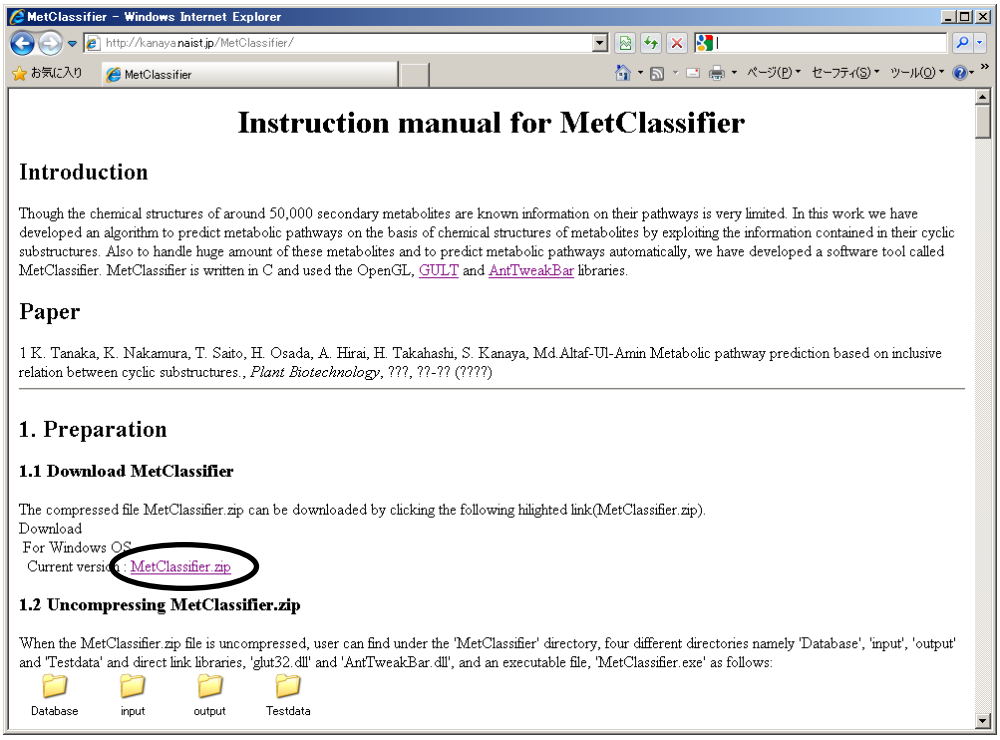


図 112 圧縮ファイルのダウンロード

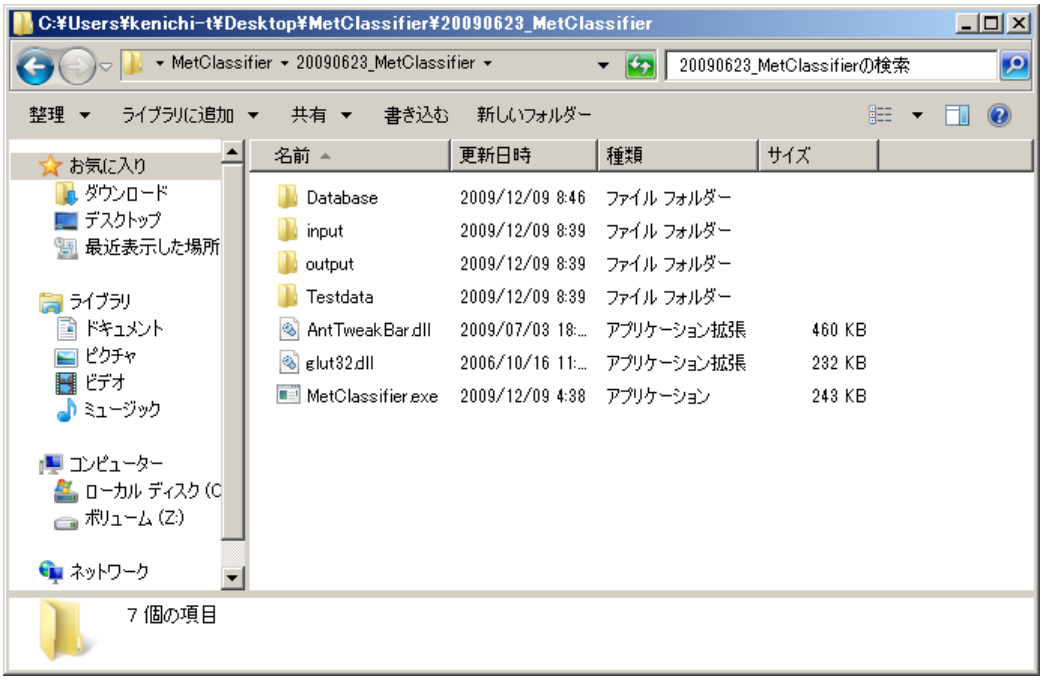


図 113 圧縮ファイルの中身

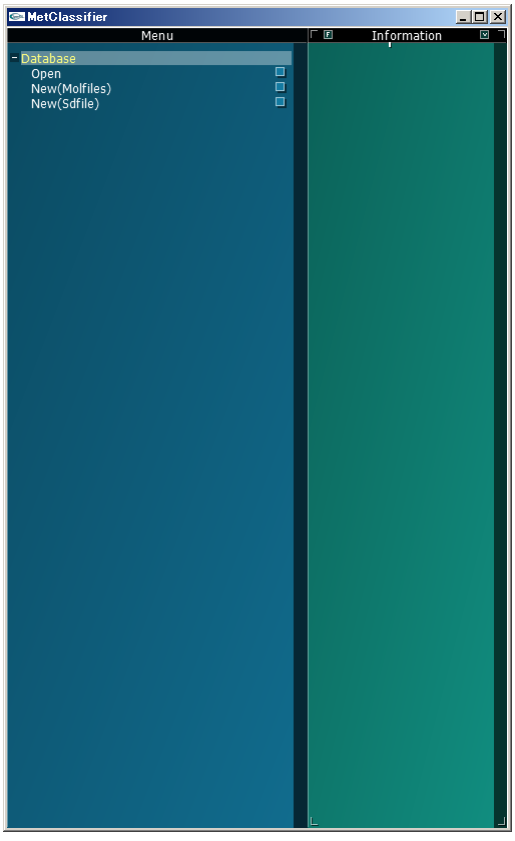


図 114 MetClassifier メインウィンドウ

第11.2.2項　　化合物データベースを開く

メインウィンドウから Database→Open を選択し、ダウンロードしたファイルの Database フォルダ内の「KNApSAcK_20091030.dat」を選択すると[12]（図 115）。メインウィンドウにメニューが追加され、Compound ウィンドウが開く（図 116）。

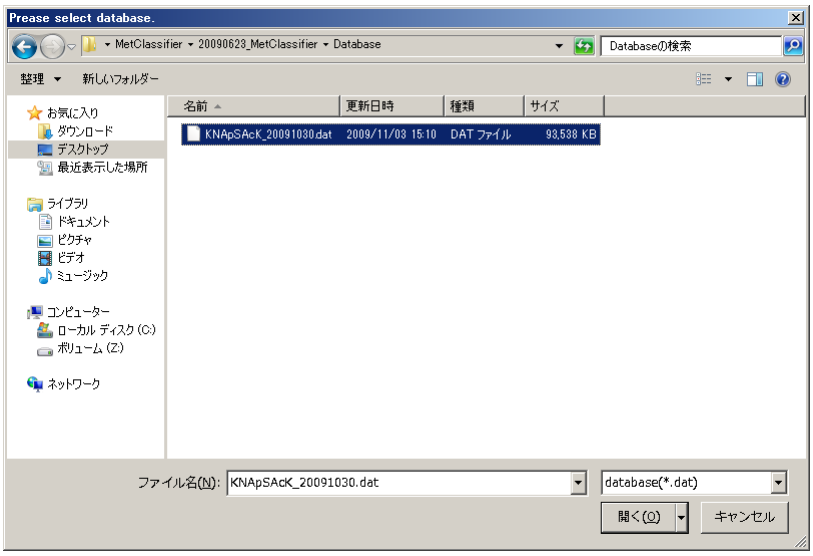


図 115 データベースの選択

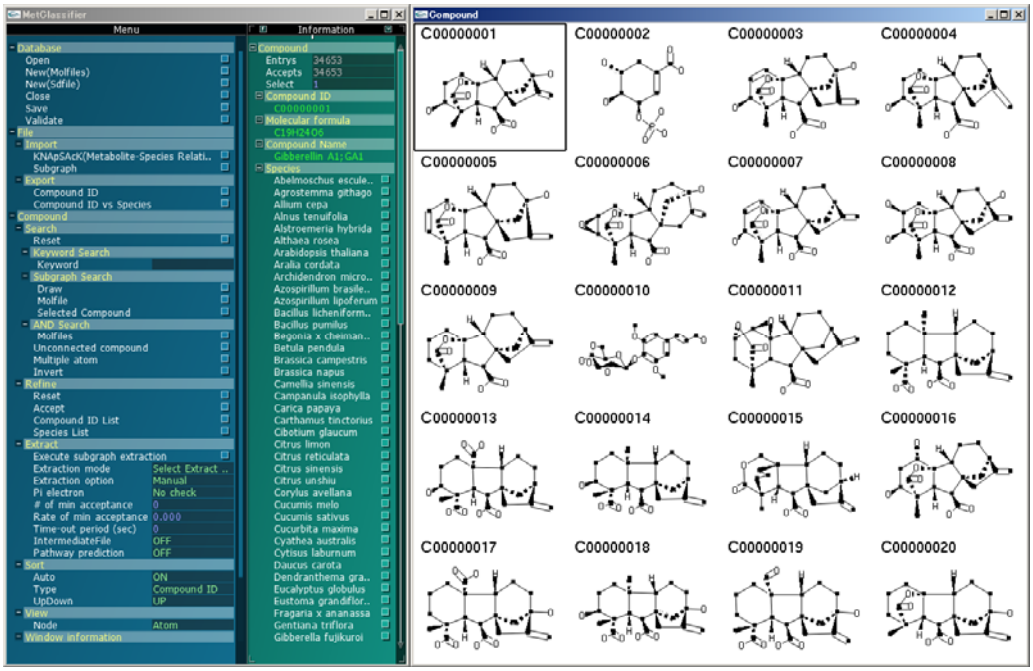


図 116 データベースを開いた状態

[12] 同封されているデータファイルは時期によって異なる。

第11.2.3項　　化合物データベースの絞り込み

Compound ウィンドウ上の化合物をクリックすると、連動してメインウィンドウの化合物情報が更新される（図 117）。今回は、*Camellia sinensis* に関する化合物に絞り込みたいので、C00000001 を再度クリックする。メイン画面化合物情報の Species の項目に「*Camellia sinensis*」の項目があるのでクリックすると、「*Camellia sinensis*」の保有する化合物の検索が行われる。Compound ウィンドウにおいて矢印キー、Page up キー、Page Down キーを用いることで、選択化合物を切り替えることが出来る。

ここで、化合物ウィンドウの化合物の境界にマウスカーソルを移動させると、カーソルの形状が変化する（図 118）。この状態でドラッグを行うことで、拡大表示（図 119）、縮小表示（図 120）が可能である。

メインウィンドウのメニュー「Compound Window→Refine→Accept」を押すと、検索が確定され、データベースが絞り込まれる（図 121）。

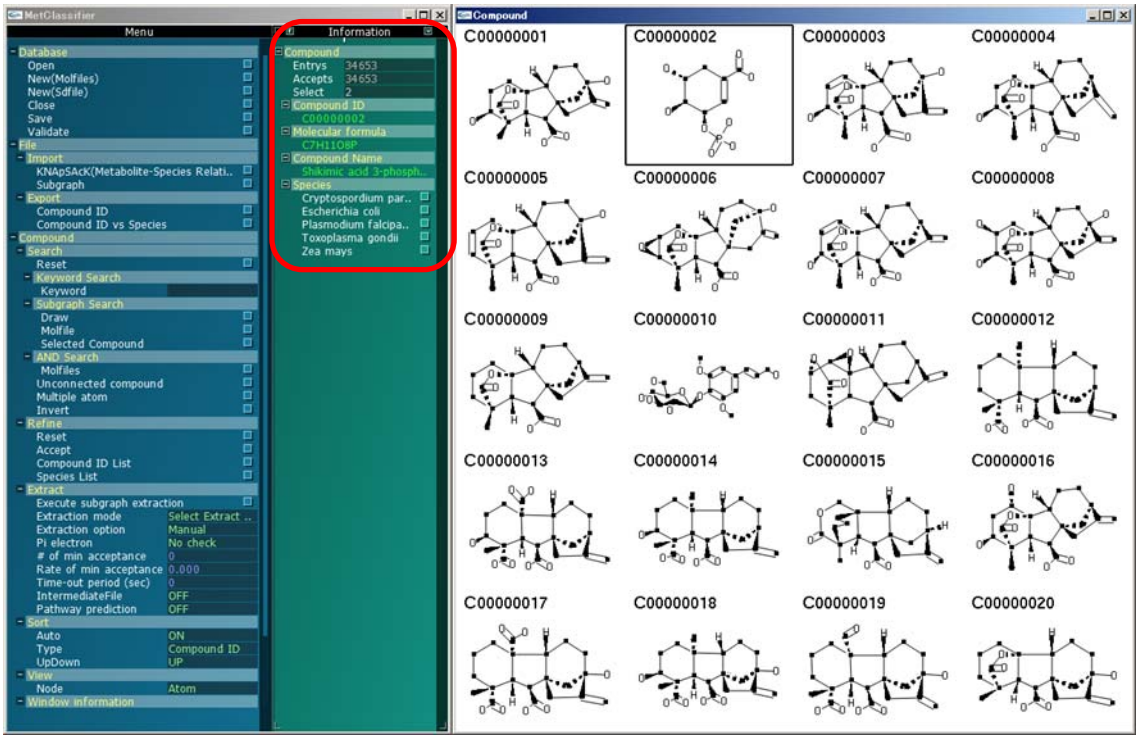


図 117 検索に連動して化合物情報が更新される

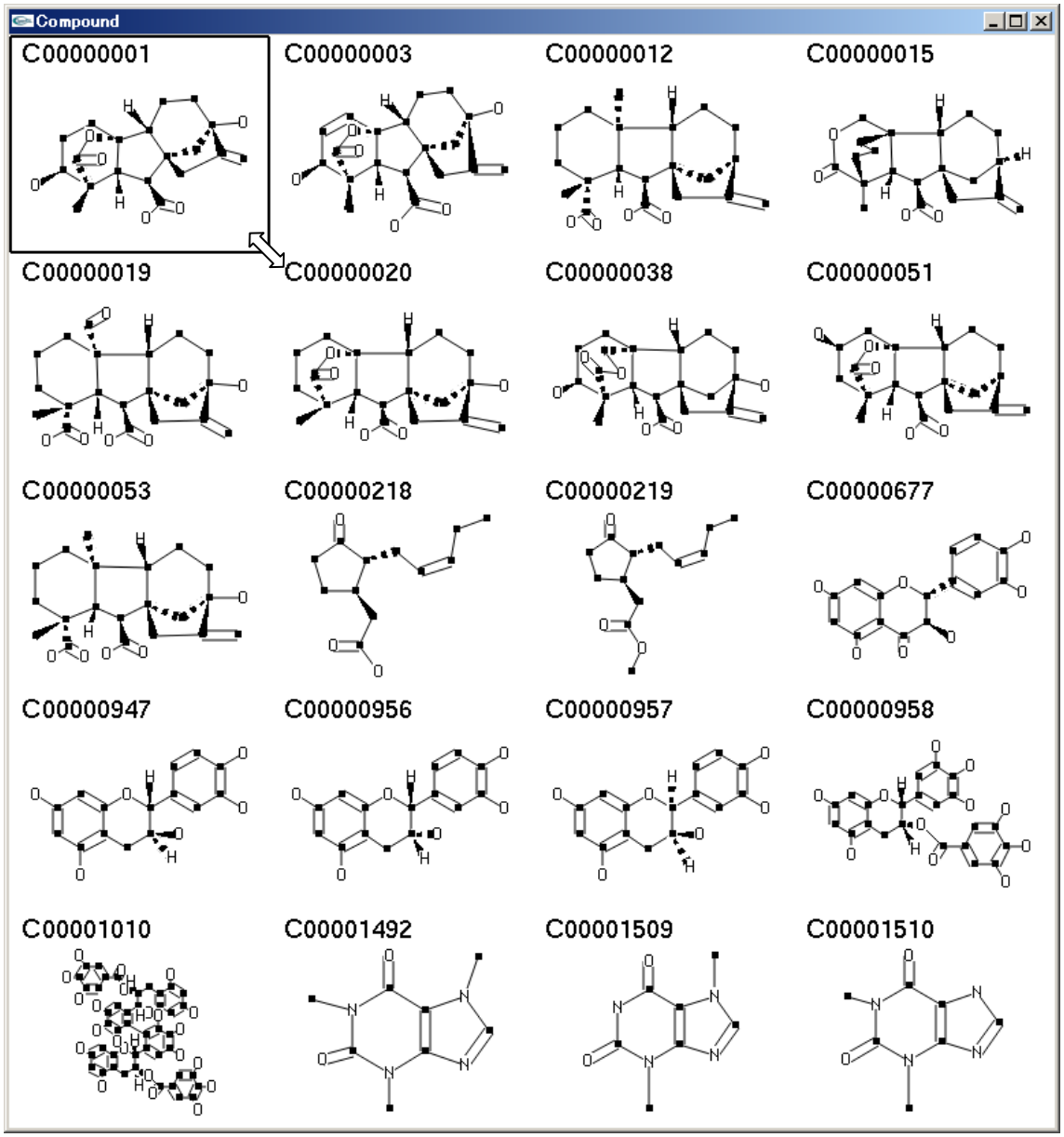

Compound
C00000001
C00000003
C00000012
C00000015
C00000019
C00000020
C00000038
C00000051
C00000053
C00000218
C00000219
C00000677
C00000947
C00000956
C00000957
C00000958
C00001010
C00001492
C00001509
C00001510

図 118 サイズ変更前

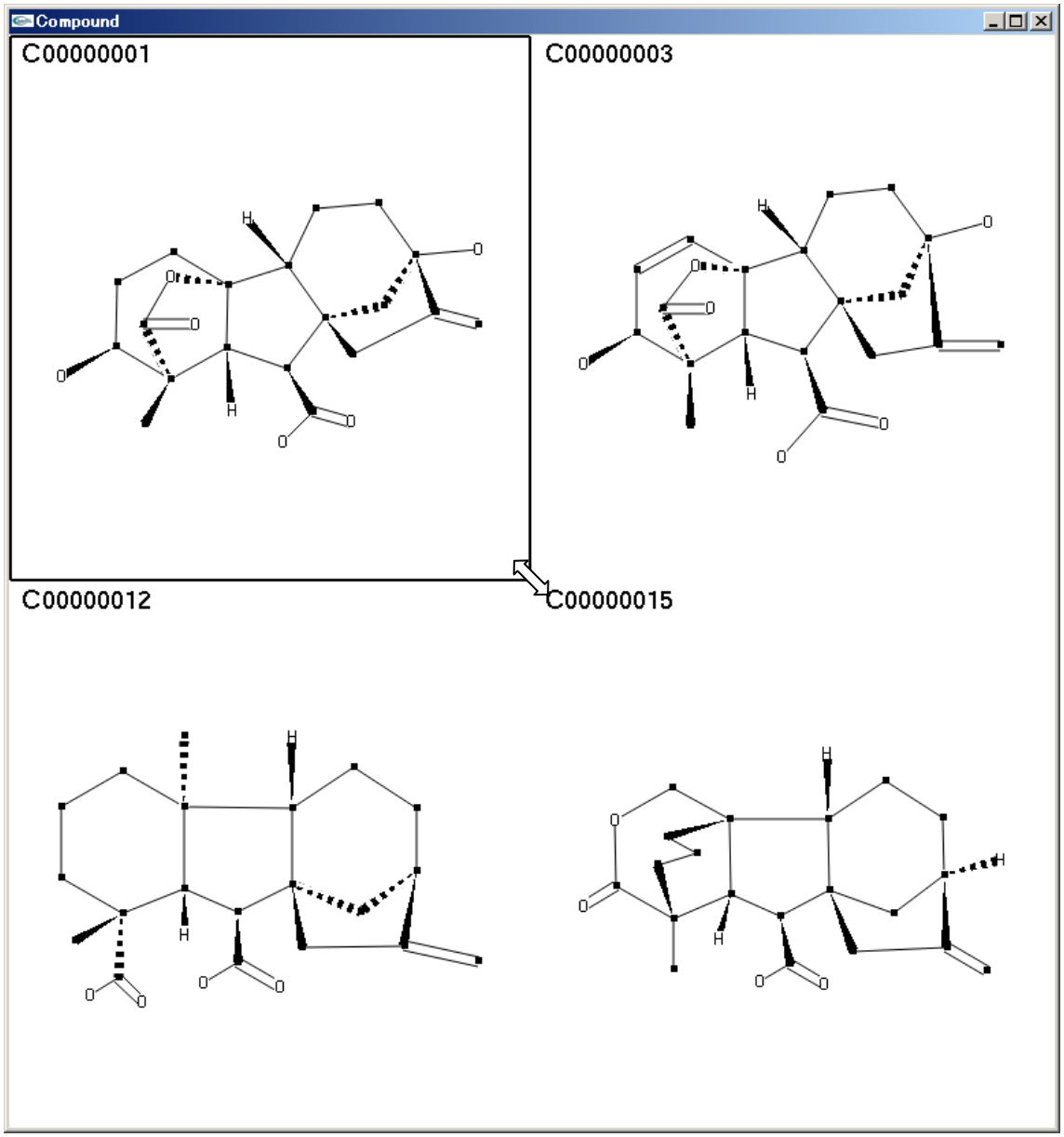


図 119 拡大表示

図 120 縮小表示

背景色がグレーの化合物は検索でヒットしなかった化合物

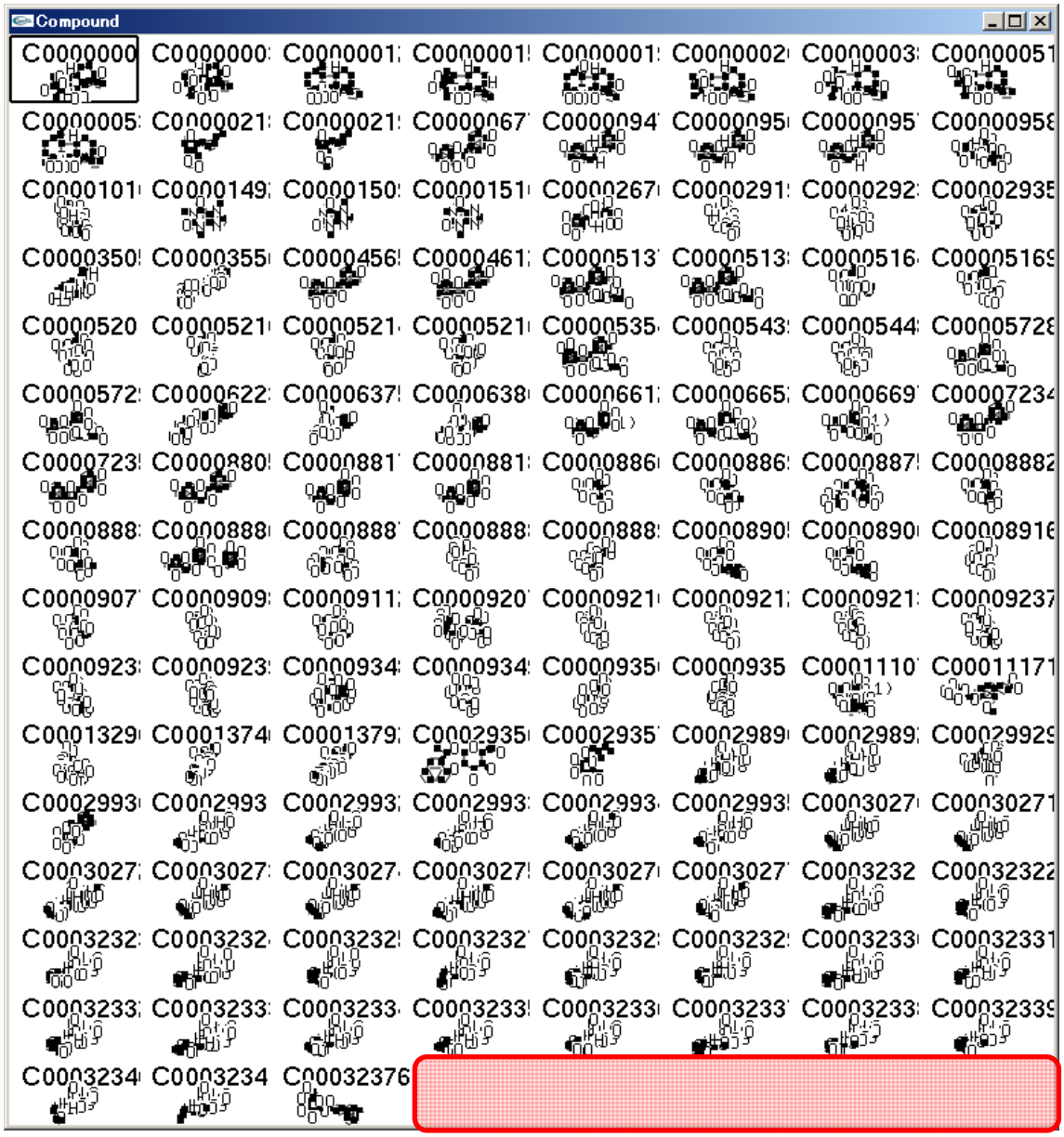


図 121 データベース絞り込みの確定

検索でヒットしなかった化合物がデータベースから取り外される。

第11.2.4項　　Subgraph の抽出

化合物データベースを絞り込んだ後、Subgraph の抽出を行う。MetClassifier では、Framework、Maximum Frequent Connected subgraph、Framework-based MFCS の 3 種類の Subgraph を抽出することが可能である。メインウィンドウの「Compound Window →Extract→Extraction mode」から抽出モードを選択すると、自動的に各設定に対する初期パラメータが設定される。検索を行う際にπ電子を考慮にいれるかどうか、最小受理数、最小受理率、制限時間、同時に経路予測を行うかどうかを設定できる。

今回は Framework-based MFCS を抽出した後、経路予測を同時に行うので「Compound Window→Extract→Extraction mode」から「Framework-based MFCS」を選択し（図 122)、「Compound Window→Extract→Pathway prediction」から「Inclusive relation」を選択した後（図 123)、「Compound Window→Extract→Execute subgraph extraction」を選択する（図 124)。

MetClassifier 起動時に同時に立ちあがるコマンドプロンプトで処理がどこまで進んだのかを確認することが出来る（図 125)。今回のケースだと 20 分程度で経路予測が完了する。

図 122 Subgraph 抽出モードの決定

図 123 同時に代謝経路予測を行う場合は設定

図 124 Subgraph 抽出処理の実行

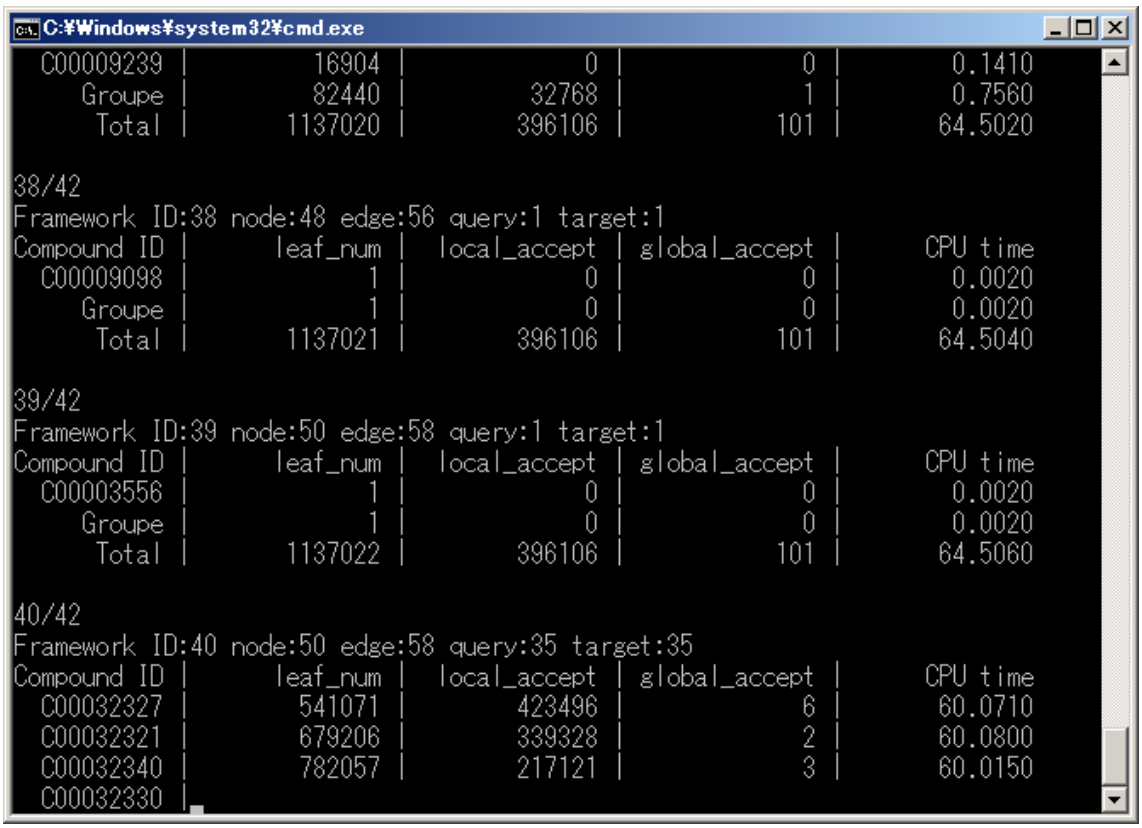


図 125 途中経過の確認

左から処理対象の Compound ID、探索木に表れた Subgraph の数、局所的な比較で MFCS の候補となった Subgraph の数、大域的な比較で MFCS であることが確定した Subgraph の数、処理時間が表示される。

第11.2.5項　　予測結果に対する各種操作

経路予測が完了すると、Subgraph ウィンドウ（図 126）および Pathway ウィンドウ（図 127）が表示される。Pathway ウィンドウは初期状態では 3 次元の階層表示となっている。Pathway ウィンドウ上でマウス右ボタンを使いドラッグを行うと、視点を回転させることが可能である。画面上のノードを選択すると、選択したノードは緑に、Ancestore subgraph に対応するノードは青に、Descendent subgraph に対応するノードは赤で表示される（図 128）。Subgraph ウィンドウおよび Pathway ウィンドウ上の Subgraph をクリックすると連動して選択した Subgraph をクエリとした部分構造検索が実行され、Compound ウィンドウが更新される。

Pathway ウィンドウ操作に関連するメニューは、Initialize、Layout mode、Dimension、Calculation、Alignment、Isomorphic、Node mode、Cell size、Gain、Attenuation が存在する（図 129）。

Initialize は描画の初期化を行う。

Layout mode には Hierarchy、Spread、Random の 3 タイプがある。Hierarchy を選択した場合、垂直方向の座標は入力 Subgraph の包含関係から得た階層情報で固定し、水平方向および奥行方向の座標は Network Self-Organization によって計算を行う。Spread を選択した場合、全方向の座標を Network Self-Organization によって計算する。Random を選択した場合、座標をランダムに設定する（図 130）。

Dimension では描画エリアの領域を 2 次元と 3 次元から選ぶことが出来る。3 次元を選んだ場合は、画面の幅が奥行きの幅になる。視点の回転を考慮にいれて、描画エリアは画面幅を直径とした円柱状にしている（図 131）。

Calculation は、座標計算の停止と実行を切り替える。

Alignment は Hierarchy の 2 次元表示の場合に有効な機能である。Alignment を押すと、化合物が階層ごとに均等に配置される（図 132）。

Isomorphic を選ぶと、Node mode で Subgraph を選択した際に、入力代謝物と同形がとれている部分構造のみが化学構造で表示され、中間構造として抽出された部分構造は Dot で表示される。また、Network Self-Organization を実行中に、最近隣にいるノードの対象が同形のとれた部分構造のみになる。これによって、同形のとれた部分構造に割り当てられる空間が広がる。

Node mode では、Dot、Subgraph ID、Subgraph の 3 タイプを選ぶことが出来る（図 133）。Dot を選ぶと、ネットワーク上で部分構造は点で表される。この時、同形の取れている部分構造は大きな Dot で表示される（図 134）。Subgraph ID を選択した場合は、Subgraph ID が表示される。同形の取れている Subgraph の ID には大きなフォントが採用される（図 135）。Subgraph を選択すると、Subgraph が表示される。同形のとれている化合物には枠がつく（図 136）。

Cell size では Subgraph 表示の際に化学構造の描画サイズを設定できる。

Gain と Attenuation は Network Self-Organization で用いている計算パラメータである。

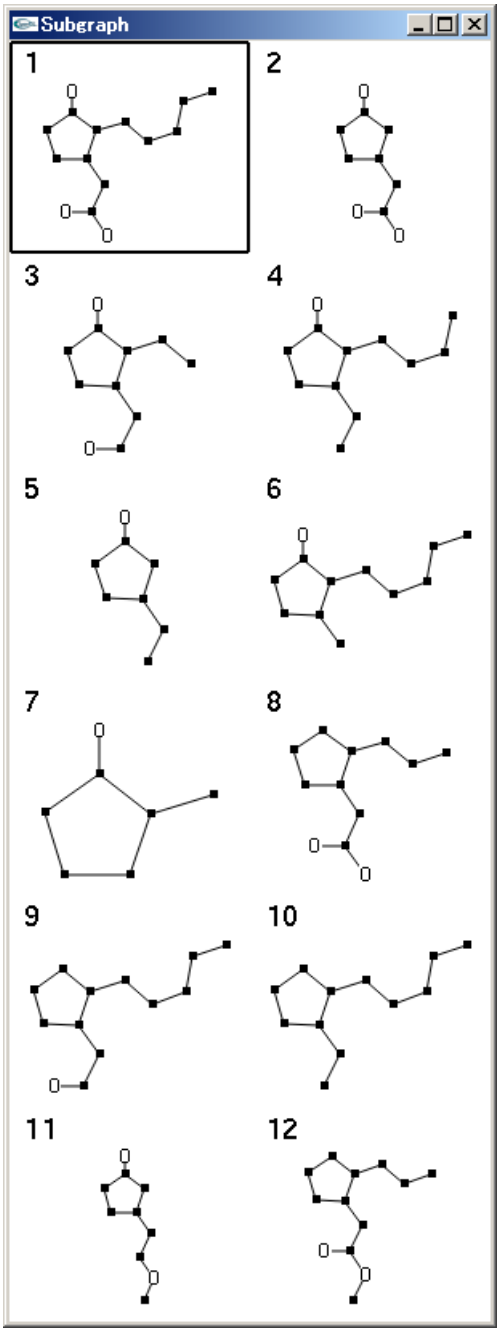


図 126 抽出した Subgraph

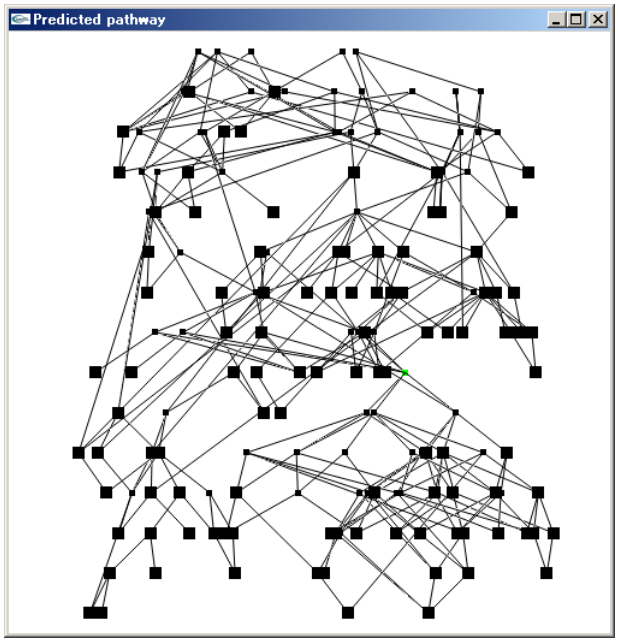


図 127 Pathway ウィンドウ（3D 階層表示）
1 分程度放置した状態、マウス右ボタンを用い、ドラッグすることで視点の回転が出来る。

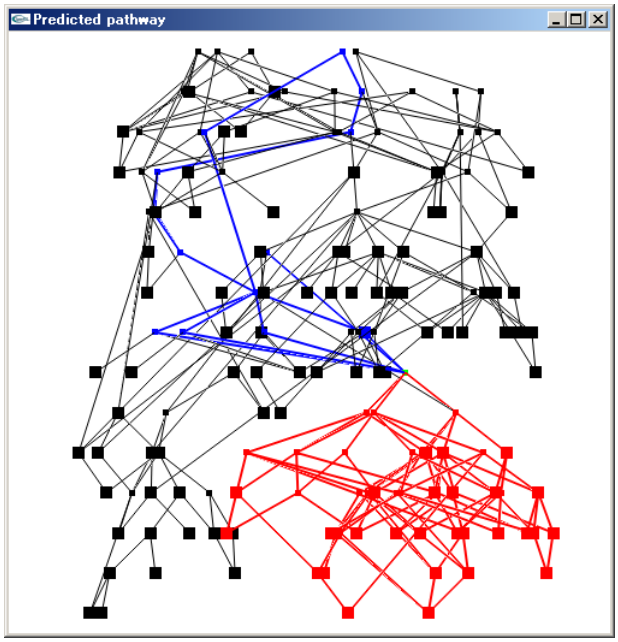


図 128 ノードを選択した状態
選択したノードは緑に、Ancestore subgraph に対応するノードは青に、Descendent subgraph に対応するノードは赤で表示される。

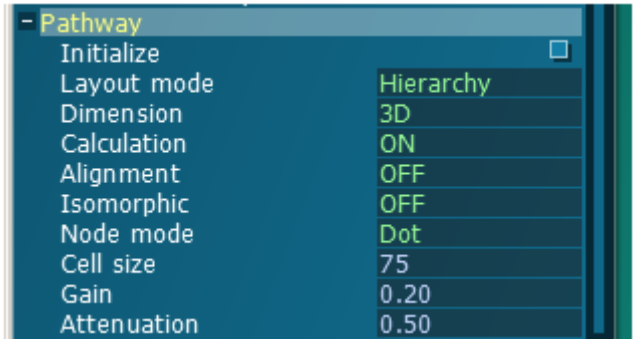


図 129 Pathway ウィンドウ操作関連メニュー

図 130 描画モード

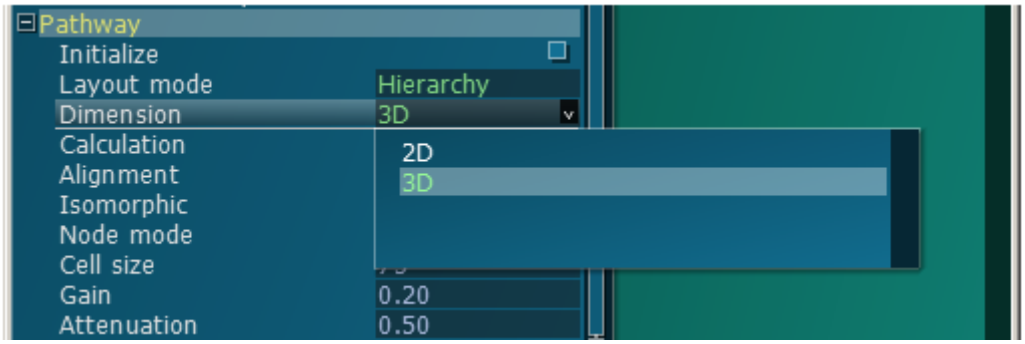


図 131 描画モード（描画領域の設定）

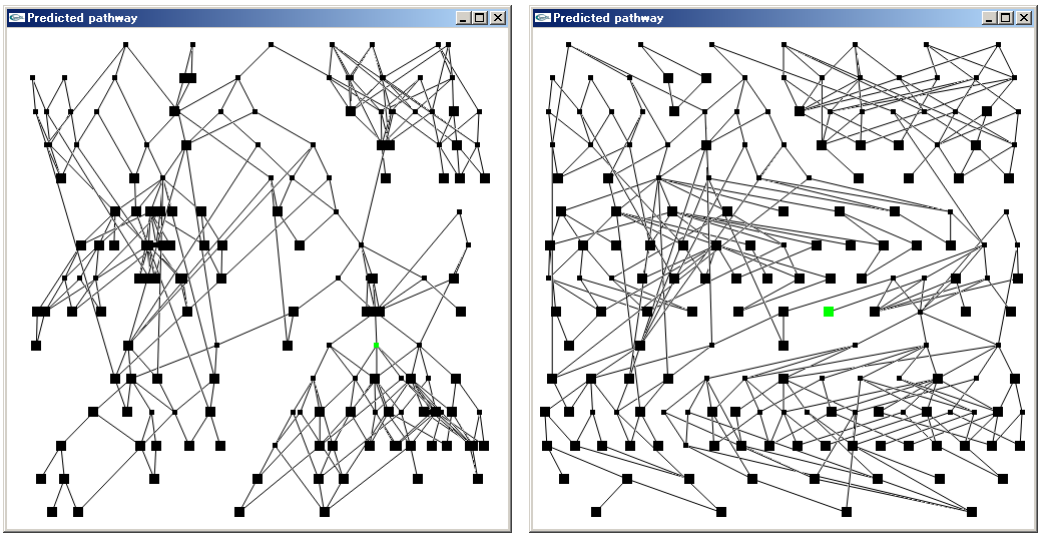


図 132 Alignment

階層ごとに均等に配置される。

図 133 描画モード（Dot、Subgraph ID、Subgraph）

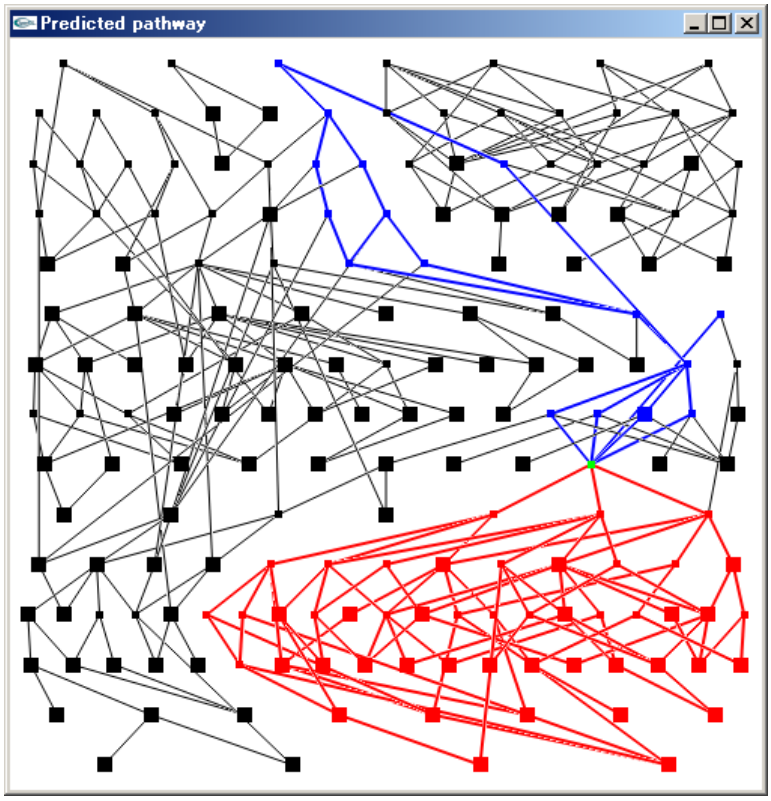


図 134 Dot で表示

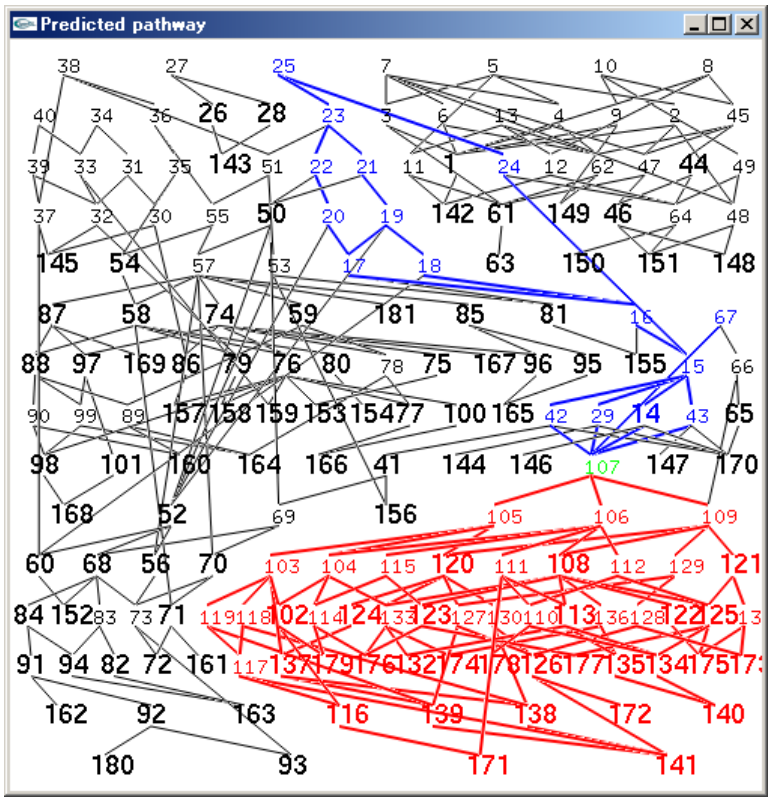


図 135 Subgraph ID で表示

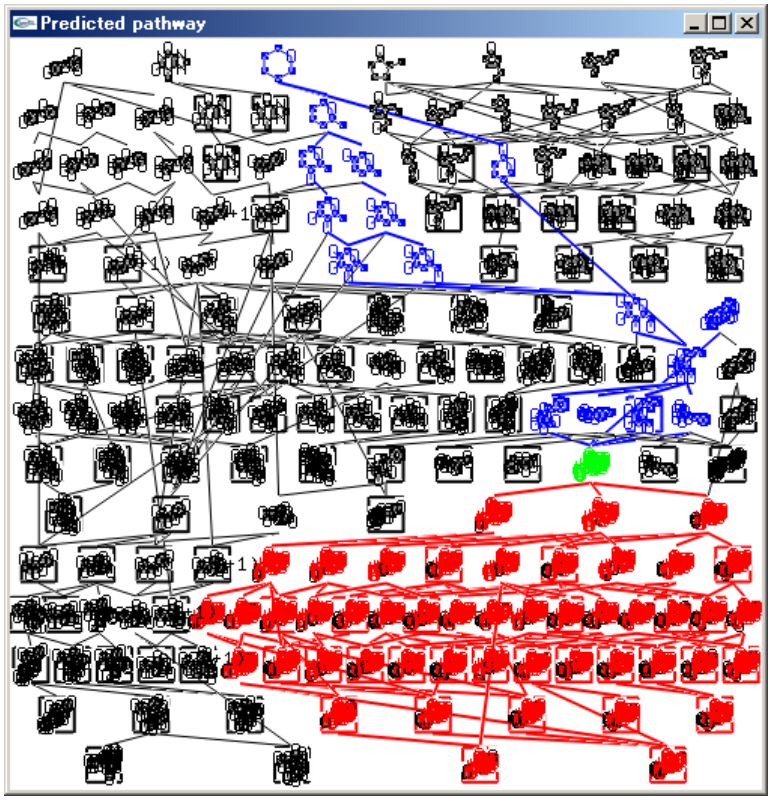


図 136 Subgraph で表示

第11.2.6項　　各種データの入出力

MetClassifier は以下のファイル入出力機能を実装している。

ファイル入力メニューは KNApSAcK(Metabolite-Species Relation)、Subgraph、Isomorphism(Compound ID vs Subgraph ID)、Subgraph level pathway、Coordinate が存在する。

KNApSAcK(Metabolite-Species Relation)では、KNApSAcK の代謝物-生物種関連情報ファイルを読み込む。読み込んだ後にデータベースを作成することで、データベースには生物種と代謝物の関連情報が保存された状態で保存される。

Subgraph、Isomorphism(Compound ID vs Subgraph ID)、Pathway、Coordinate は事前に出力した対応ファイルを読み込む[13]。

ファイル出力メニューは Compound ID、Compound ID vs Species、Subgraph、

[13] Subgraph に関しては、元になった化合物と Subgraph に含まれる原子と結合の内部 ID から構成される為、異なるデータベースから出力された Subgraph ファイルは読み込めない。原子と結合の内部 ID はデータベース作成時に用いた Molfile に格納されている原子情報と結合情報の順が採用される。

Isomorphism(Compound ID vs Subgraph ID)、Subgraph level pathway、Coordinate、Direct Pathway、Shortest Pathway が存在する。

Compound ID は Compound ウィンドウに表示されている化合物の Compound ID を出力する（図 138)。

Compound ID vs Species は Compound ウィンドウに表示されている化合物に対して、どの化合物をどの生物種が保有しているのかをマトリックスで出力する（図 139)。

Subgraph は抽出した Subgraph を抽出元の化合物の Compound ID、化合物の原子の内部 ID、結合の内部 ID の情報を出力する（図 140)。

Isomorphism (Compound ID vs Subgraph ID)は、Compound と同形のとれている Subgraph の ID をタブ区切り形式で出力する（図 141)。

Subgraph level pathway は予測した代謝経路に含まれる Parent subgraph と Child subgraph の Subgraph ID をタブ区切り形式で出力する（図 142)。

Coordinate は Pathway ウィンドウ上における各 Subgraph の座標を出力する。Subgraph ID、x 座標、y 座標、z 座標の情報をタブ区切り形式で出力する（図 143)。

Direct pathway および Shortest pathway は、予測した代謝経路に含まれる、化合物 1 の Compound ID、化合物 2 の Compound ID、最短経路長、代謝経路に出現する Subgraph ID（化合物 1-中間構造-化合物 2）をタブ区切りで出力する。入力化合物と同形のとれている Subgraph ID には( )が付く（図 144、図 145)。

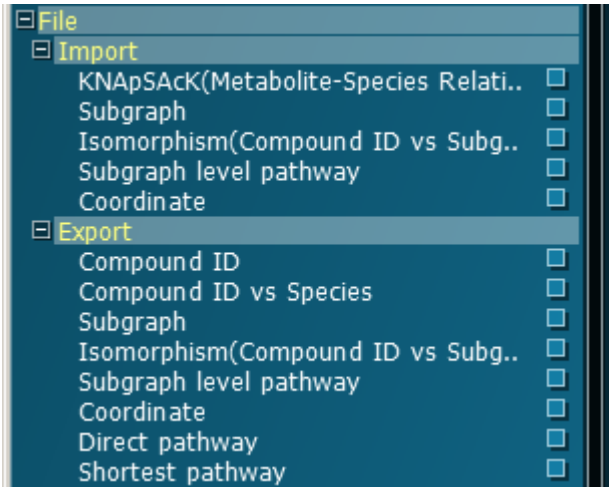


図 137 ファイル入出力操作関連メニュー

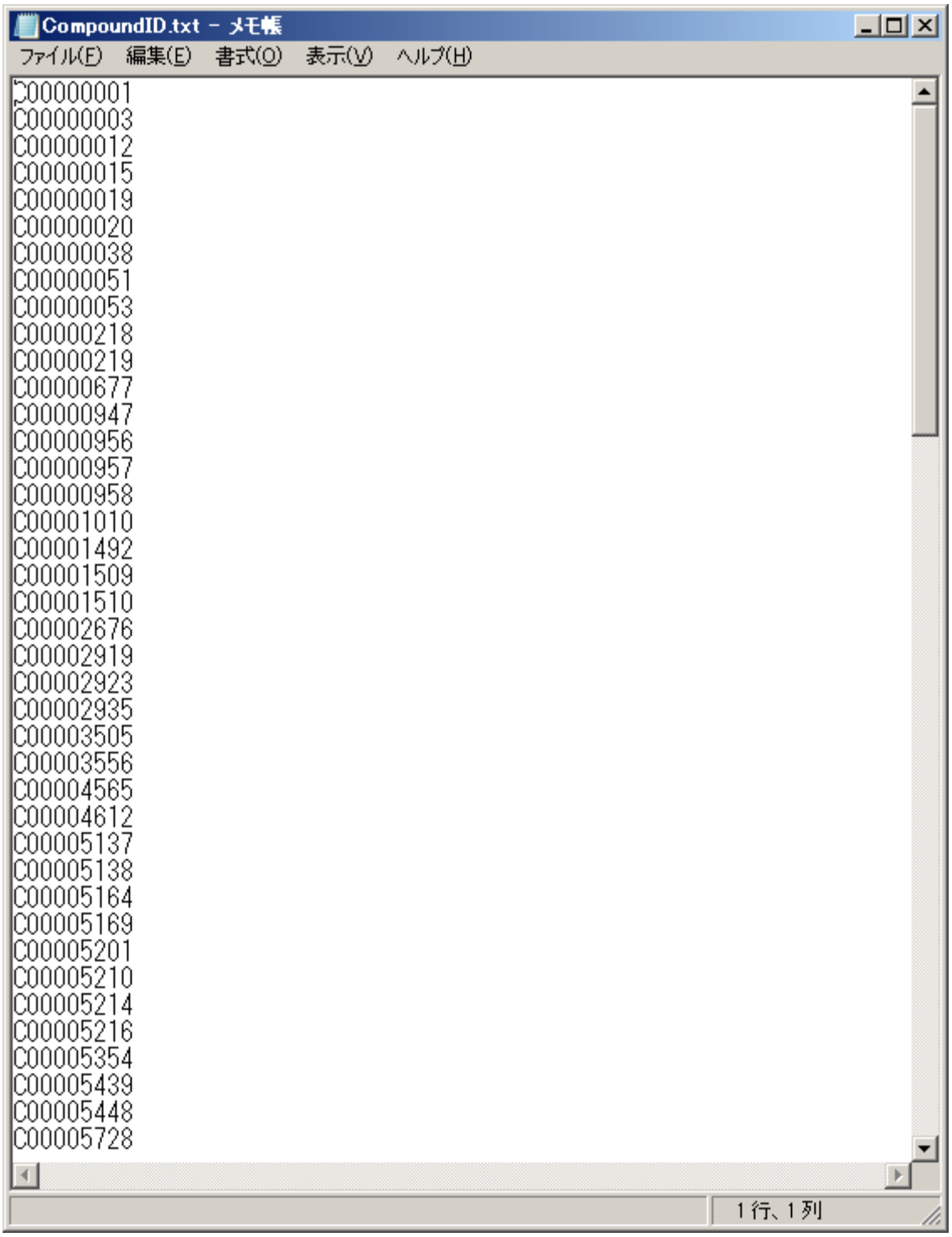
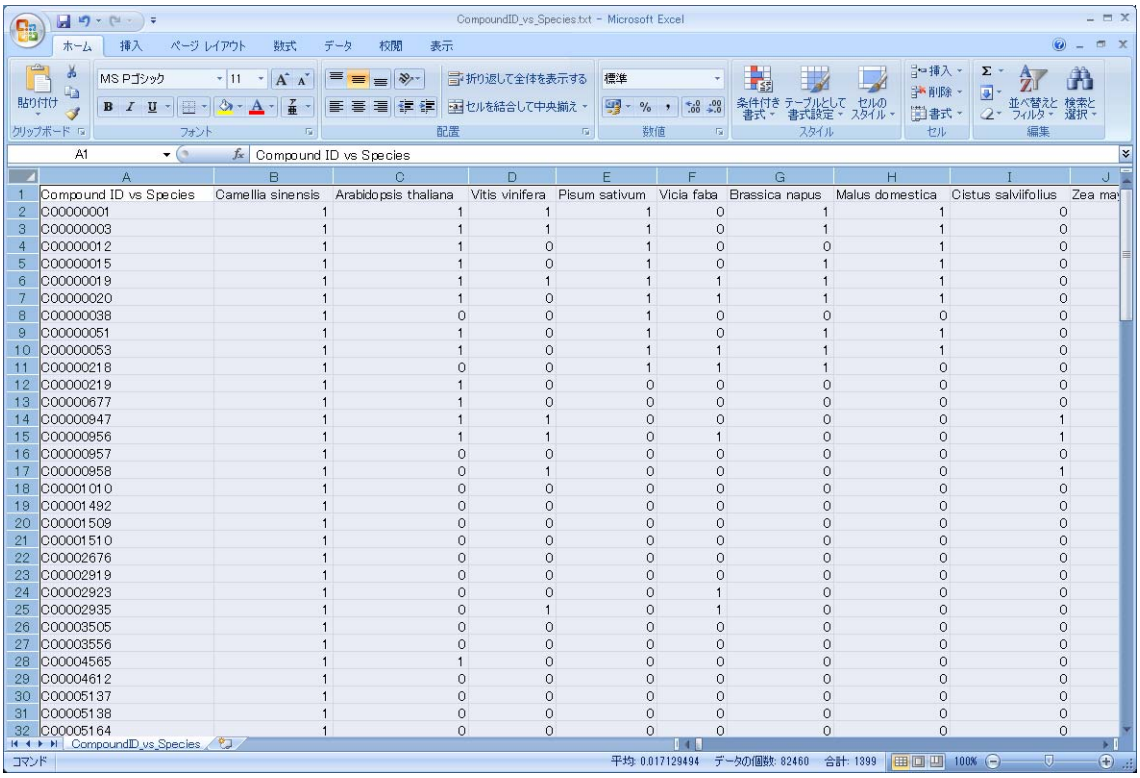


図 138 Compound ID

出力内容

化合物 ID



	A	B	C	D	E	F	G	H	I	J
1	Compound ID vs Species	Camellia sinensis	Arabidopsis thaliana	Vitis vinifera	Pisum sativum	Vicia faba	Brassica napus	Malus domestica	Cistus salvilifolius	Zea ma
2	C00000001	1	1	1	1	0	1	1	0	
3	C00000003	1	1	1	1	0	1	1	0	
4	C00000012	1	1	0	1	0	0	1	0	
5	C00000015	1	1	0	1	0	1	1	0	
6	C00000019	1	1	1	1	1	1	1	0	
7	C00000020	1	1	0	1	1	1	1	0	
8	C00000038	1	0	0	1	0	0	0	0	
9	C00000051	1	1	0	1	0	1	1	0	
10	C00000053	1	1	0	1	1	1	1	0	
11	C00000218	1	0	0	1	1	1	0	0	
12	C00000219	1	1	0	0	0	0	0	0	
13	C00000677	1	1	0	0	0	0	0	0	
14	C00000947	1	1	1	0	0	0	0	1	
15	C00000956	1	1	1	0	1	0	0	1	
16	C00000957	1	0	0	0	0	0	0	0	
17	C00000958	1	0	1	0	0	0	0	1	
18	C00001010	1	0	0	0	0	0	0	0	
19	C00001492	1	0	0	0	0	0	0	0	
20	C00001509	1	0	0	0	0	0	0	0	
21	C00001510	1	0	0	0	0	0	0	0	
22	C00002676	1	0	0	0	0	0	0	0	
23	C00002919	1	0	0	0	0	0	0	0	
24	C00002923	1	0	0	0	1	0	0	0	
25	C00002935	1	0	1	0	1	0	0	0	
26	C00003505	1	0	0	0	0	0	0	0	
27	C00003556	1	0	0	0	0	0	0	0	
28	C00004565	1	1	0	0	0	0	0	0	
29	C00004612	1	0	0	0	0	0	0	0	
30	C00005137	1	0	0	0	0	0	0	0	
31	C00005138	1	0	0	0	0	0	0	0	
32	C00005164	1	0	0	0	0	0	0	0	

図 139 Compound ID vs Species

出力内容

化合物と生物種の関係情報をマトリックスで表示

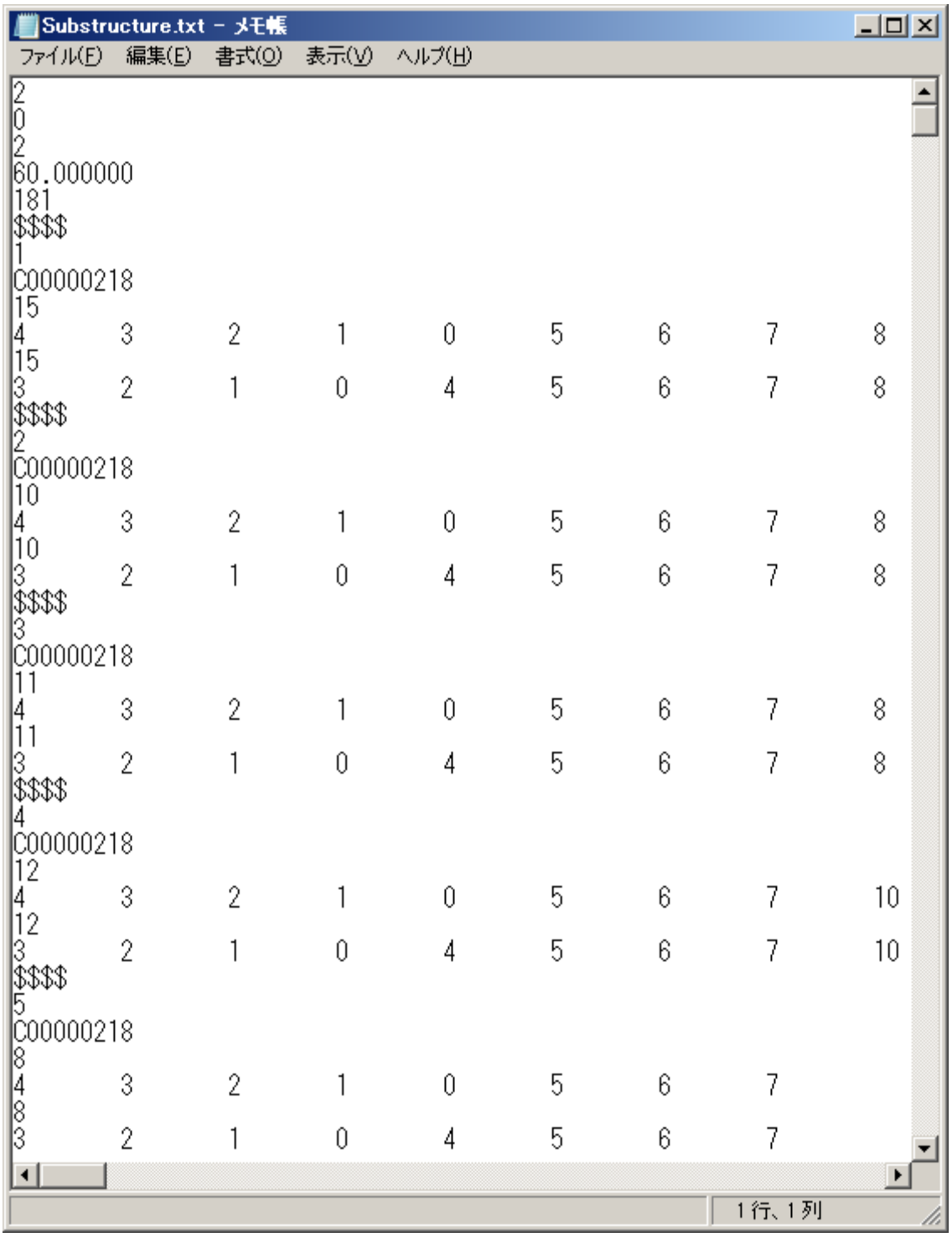


図 140 Subgraph

出力内容

Subgraph 抽出条件、抽出 Subgraph 数、Subgraph ID、Subgraph 抽出元の Compound ID、Subgraph に含まれる node 数、node ID、Subgraph に含まれる edge 数、edge ID

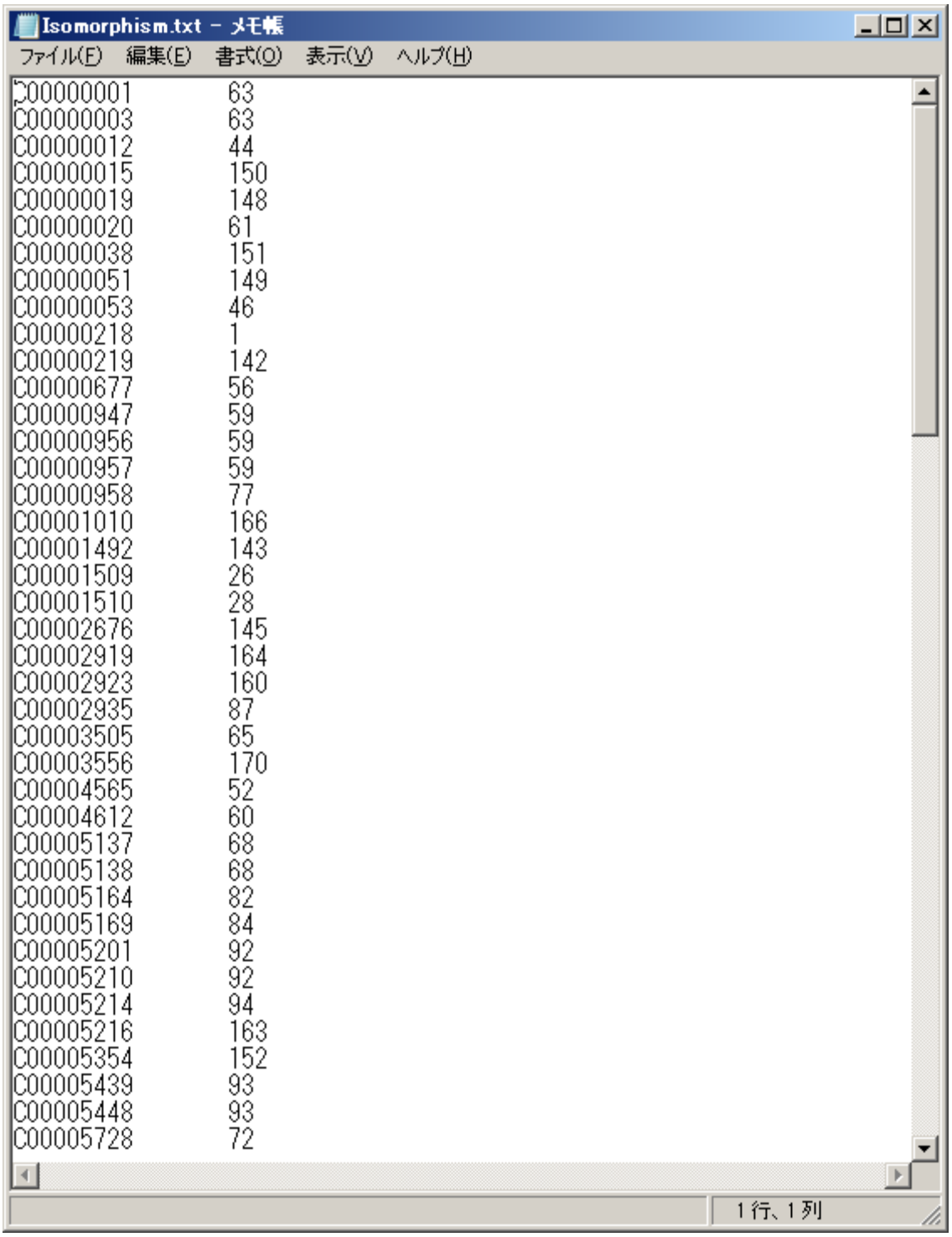


図 141 Isomorphism(Compound ID vs Subgraph ID)

出力内容

Compound ID　　Subgraph ID

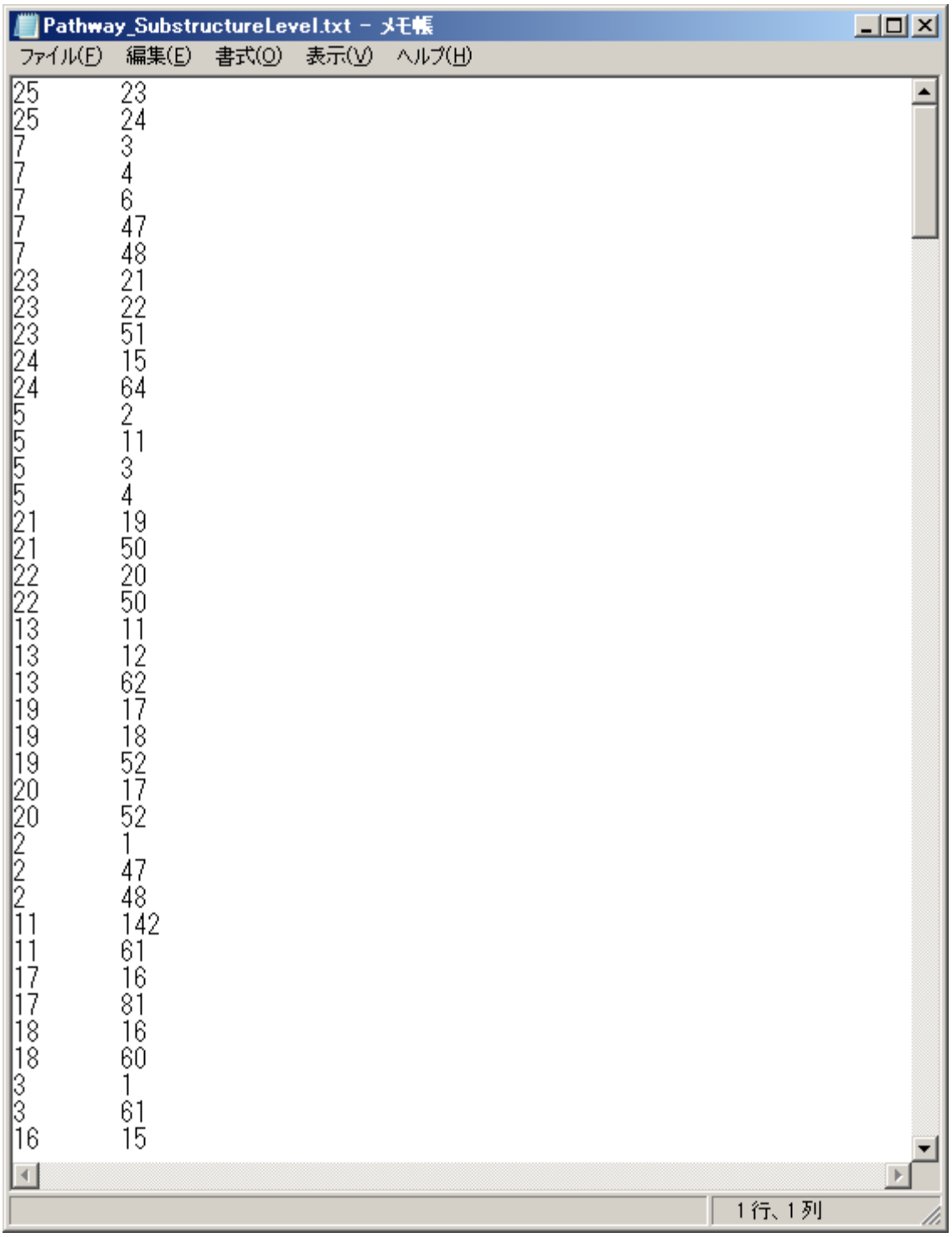


図 142 Subgraph level pathway

出力内容

Parent subgraph ID　　Child subgraph ID

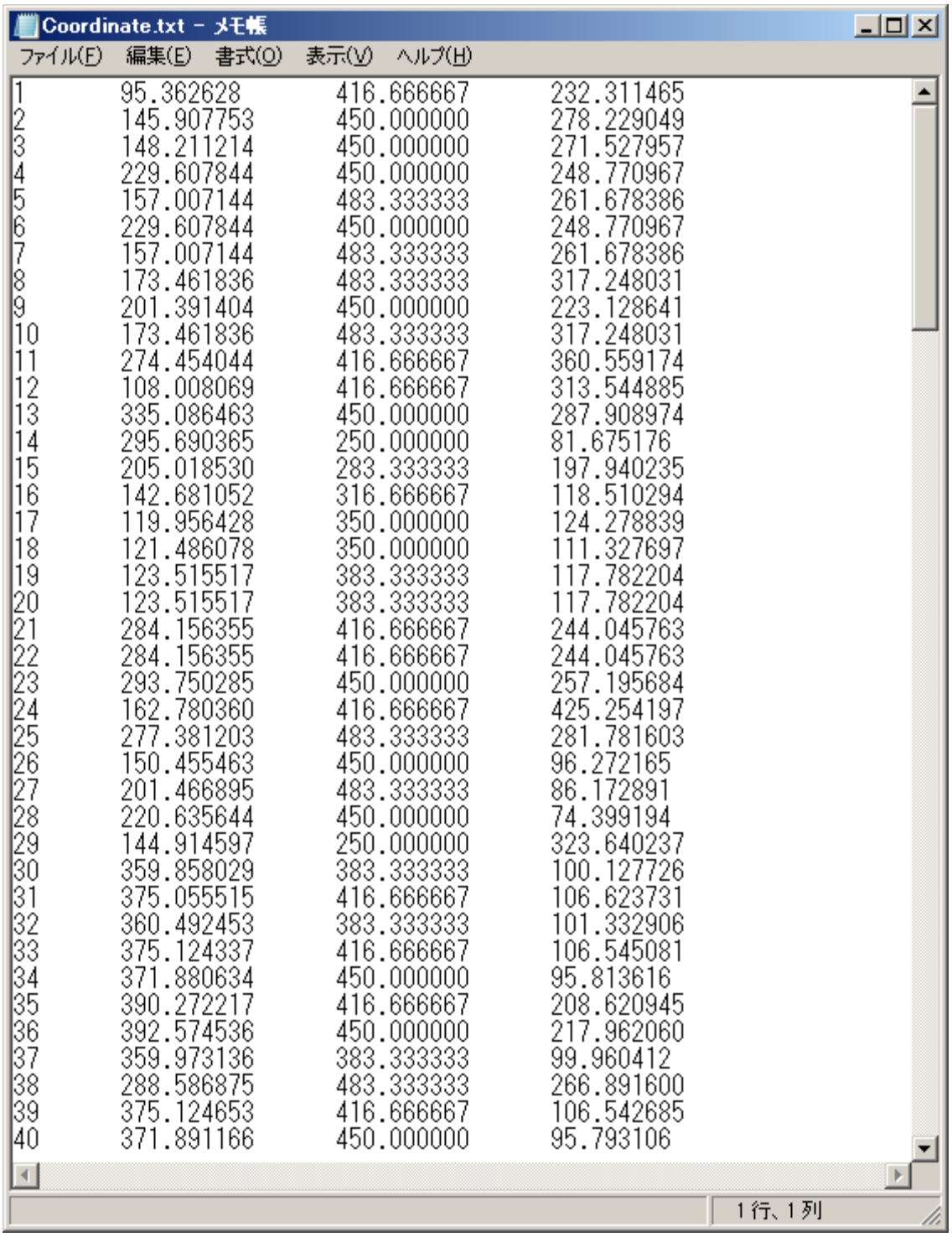


図 143 Coordinate

出力内容

Subgraph ID　　x 座標　y 座標　z 座標

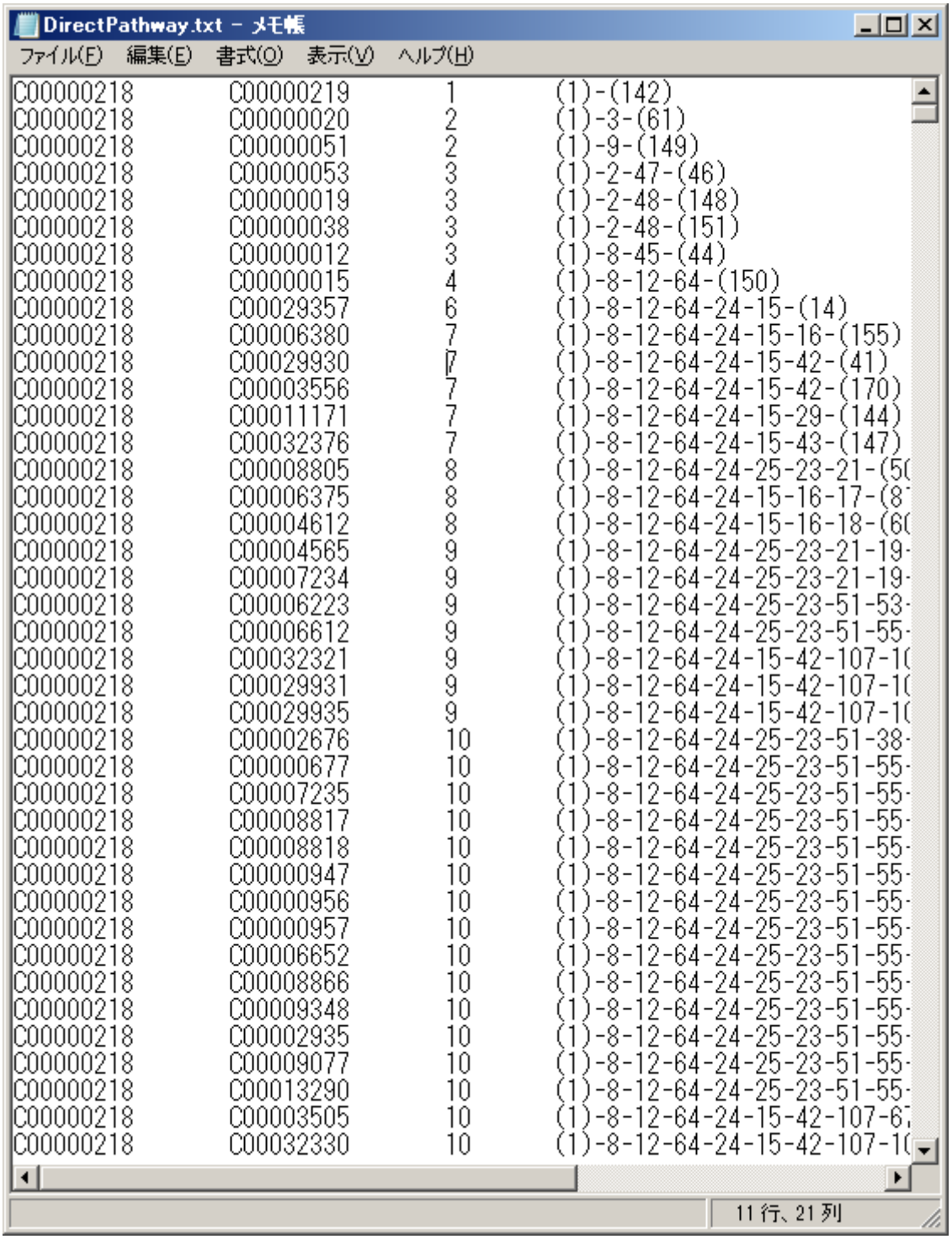


図 144 Direct pathway

出力内容

Compound ID1　Compound ID2　経路長　経由した Subgraph ID

※化合物と同形のとれている Subgraph ID は()が付く。

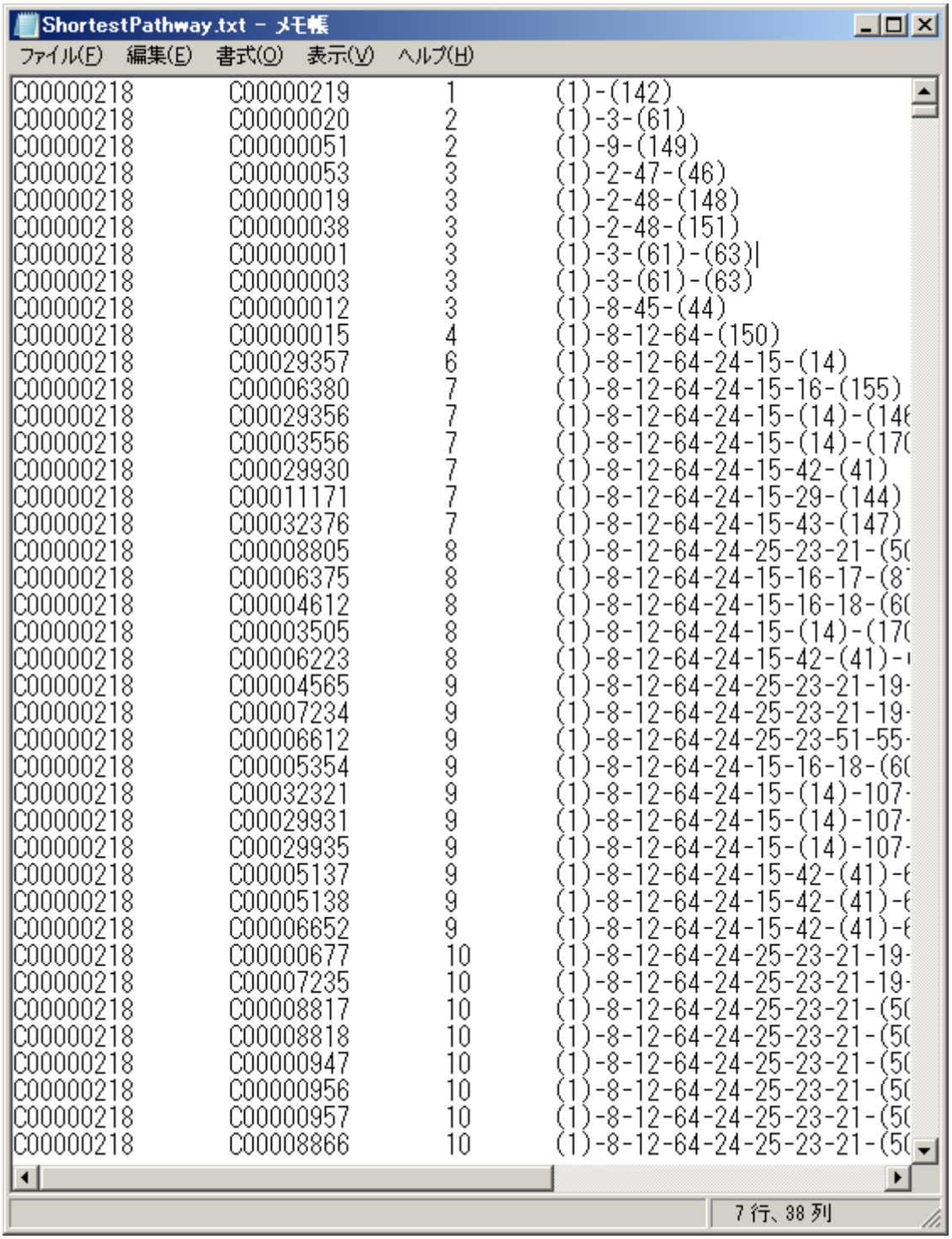


図 145 Shortest pathway

出力内容は Direct pathway と同じである。

第12章 部分構造検索エンジンの提供

本研究で開発したシステムは、代謝経路予測以外にも化学構造を評価することを目的とした様々な分野に対して応用が可能である。本章では共同研究グループが運営している化合物データベースに本研究で開発したシステムの一部を WEB 上で動作するアプリケーションへの改良および実装を行ったので、それらについて述べる。第 12.1 節ではシステム提供に至った背景、各データベースの紹介を行う。第 12.2 節ではシステム提供の過程で行った作業内容、発生した問題に対する対応方法、新機能の開発等について述べる。第 12.3 節では各データベースで提供したシステムがどのように利用されているのかについて述べる。

第12.1節　　研究背景

世の中には有償の部分構造検索モジュールはいくつか存在するが、これらのモジュールは非常に高額な年間ライセンスを必要とする。一方、無料で公開されているモジュールも存在するが、これらのモジュールは計算速度の面において実用に耐えられない。

共同研究グループで開発運営が行われている NPEdia[http://npd.riken.jp/npedia/]や MassBank[http://www.massbank.jp/]は無料で公開されている公共のデータベースサービスである。NPEdia は理化学研究所長田抗生物質研究室で運営されている、化合物の構造データ、単離・合成法、生物活性情報などをまとめた化合物データベースである。連動している天然化合物バンク NPDepo を通じて、放線菌や糸状菌から単離した代謝化合物の他、植物などから単離された二次代謝物など、多様性のある化合物の収集・保管および希望者への提供も行われている。MassBank は慶應義塾大学先端生命科学研究所（IAB） 分析化学グループ、理化学研究所植物科学研究センター（PSC）メタボローム基盤グループが中心となり、 JST-BIRD プロジェクトとして開発運営されている高分解能マススペクトルデータベースである。どちらのデータベースも、多くの研究者にとって非常に有用なサービスであり、このようなサービスを長期間にわたって運営する為には、高額なライセンス使用料を必要としない、高速に動作するシステムを構築する必要があった。

また、実装の過程において「複数の構造活性を同時に保有する化合物を検索したい」といった要望や、「MS/MS から得られる化学構造の断片情報を繋ぎ合せて元の化学構造を類推したい」という要望があった。これらを実現する為には複数のクエリ構造を入力とした複合部分構造検索機能が必要である。現状 WEB で公開されている部分構造検索サービスは単一のクエリ構造を入力としているサービスであって複数のクエリ構造を入力とするサービスは見かけないことから、これらの機能も新たに実装した。

第12.2節　　実施内容

本節では、公共データベースへのシステム提供の過程で行った作業内容、発生した問題に対する対応方法、新機能の開発等について述べる。

第12.2.1項　　公共データベースへの部分構造検索エンジンの実装

Web クライアントとして動作させる為に、システムの仕様を以下の様に決定した。

引数：Molfile 形式のクエリ構造ファイル(Dalby, 1992)
標準入力：Compound ID リスト
標準出力：部分構造検索でヒットした Compound ID

Web 画面から入力されたクエリ構造を基に生成された Molfile 形式の化学構造データを引数として用いることで部分構造検索を実行した後、標準出力に出力された Compound ID リストを用いて部分構造検索結果の Web ページを構築している（図 146)。本研究で開発したシステムが実装されるまでは、PerlMol[http://www.perlmol.org/]と呼ばれるシステムが検索エンジンとして用いられていたのだが、クエリ構造によっては 20 分以上かかっていた検索が 6 秒程度で完了するようになった。

ここまでの作業で、検索速度は従来に比べ格段に速くなったが、いくつかの問題点が残った。まず、MetClassifier では検索速度を上げる為に化合物データを事前にメモリ上に全て読み込んでいる。検索時間の 6 秒の大半はデータベースの読み込み時間であり、検索そのものにかかる時間は 1 秒未満である。検索毎にデータベースの読み込みを毎回行うのは非常に効率が悪く、サーバに対する負荷も大きい。また NPEdia に登録されている化学構造データは 4 万件近く存在し、全ての化学構造データを読み込んだ場合、約 300MB のメモリが必要となる。現在主流の 32bitCPU で動作している計算機の場合、認識できるメモリの量は 2^{32}=約 4GB である。実際には OS や他のプロセスもメモリを消費しているため、部分構造検索で用いることの出来るメモリは 4GB 以下となる。一回の検索にかかる 6 秒以内に 14 件以上の検索問い合わせが発生した場合、300M×14＝4200M　4G となりメモリを使い果たしてしまい、最悪サーバがダウンしてしまう。Web システムとして多人数の同時接続に耐えるためにはこれらの問題を解決する必要があり、次項で述べる Client-Server 型部分構造検索エンジンに改良することで対応した。

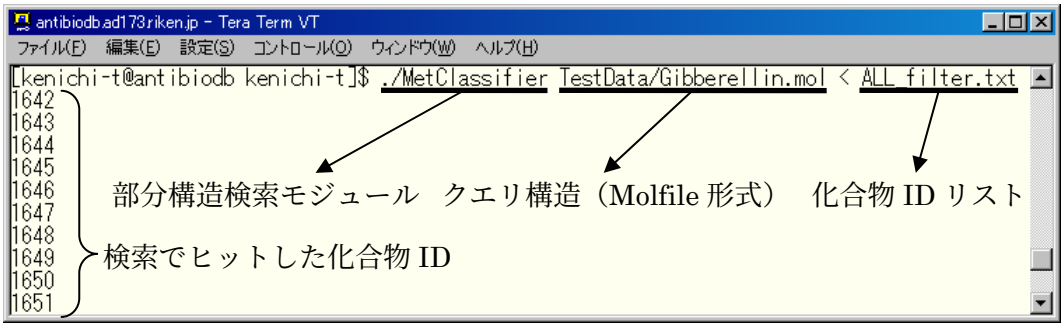


図 146 GUI 機能を取り外し CUI に改良したシステムによる部分構造検索例

Web 画面からクエリ構造を描画し検索を実行することで、部分構造検索モジュールが実行され、出力された化合物 ID を用いて検索結果出力 Web ページを作成している。ここまでの作業で、検索速度は従来に比べ格段に速くなったが、いくつかの問題点が残った。

第12.2.2項　　Client-Server 型部分構造検索エンジン

多人数の同時接続に耐えるために、Client-Server 型部分構造検索エンジンを開発した。Client および Server はそれぞれ以下の動作を行う。

Client は以下の動作を行う。

(1)Web 画面にクエリ構造が記述され検索ボタンが押されたら起動する。

(2)Client は WEB 画面から生成された Molfile を Server に転送。

(3)Server からの返答を待つ。

(4)Server から受け取った検索結果 Compound ID リストを標準出力に出力。

(5)処理を終了する。

Server は起動時に全ての化学構造データを読み込んだ後、while ループ内で以下の処理を繰り返す。

(1)Client からの通信を待つ。

(2)Client から通信があったら Molfile と検索対象 Compound ID リストを受け取る。

(3)部分構造検索を実行する。

(4)Client に検索結果 Compound ID リストを返す。

Client-Server 型に改良することで、各クエリに対する検索時間は 1 秒未満になり、同時接続時におけるメモリ使用量も減少した。この改善で外部公開を行うことで多人数からの同時アクセスが発生しても、サーバに過大な負荷がかからないようになった。

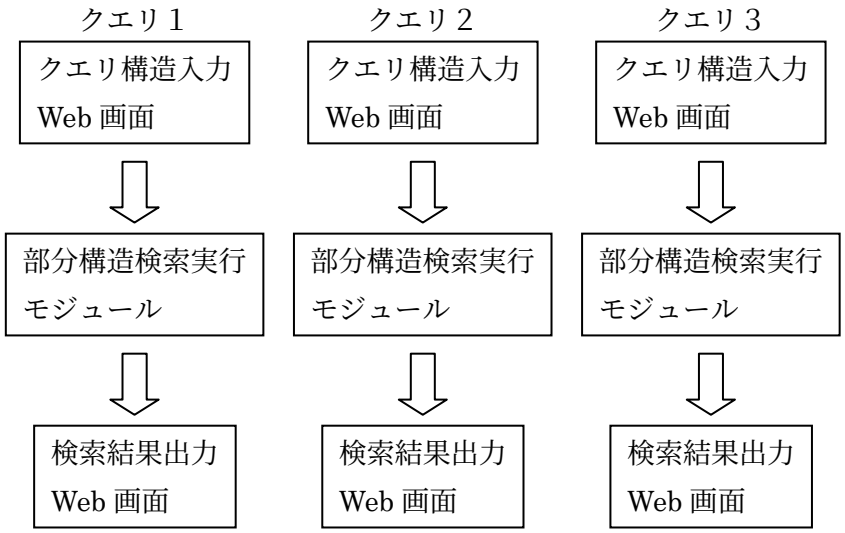


図 147 改善前のシステム

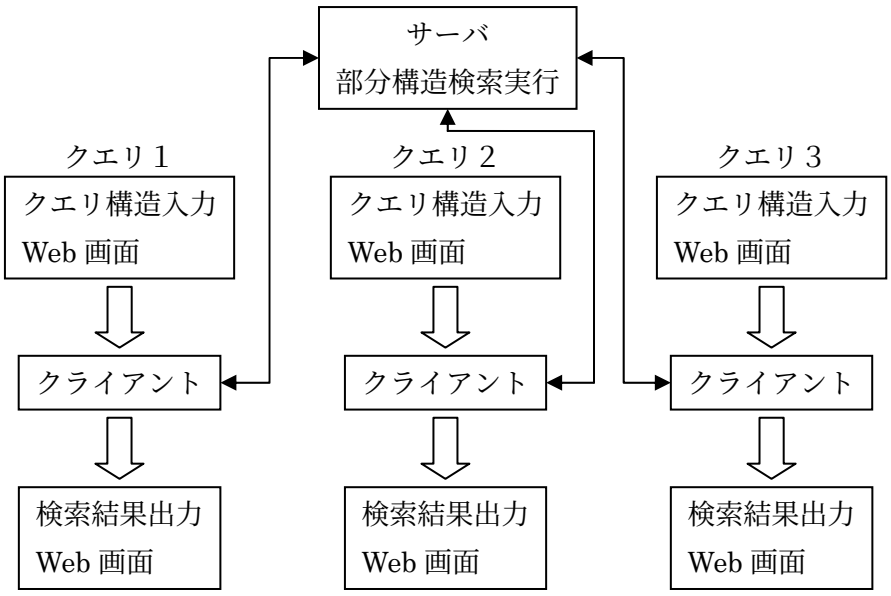


図 148 改善後のシステム

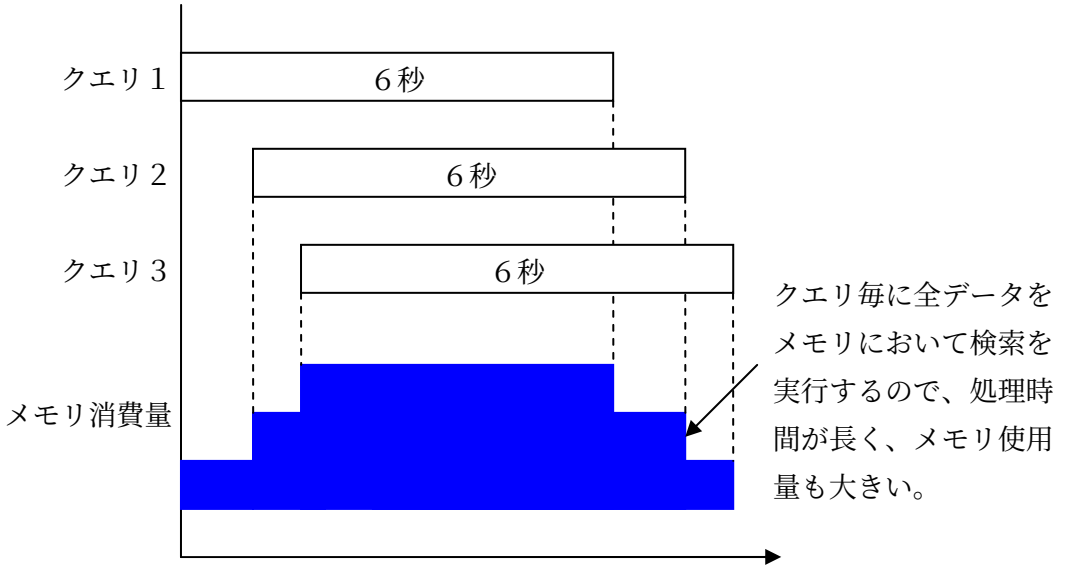


図 149 改善前の検索時間およびメモリ消費量

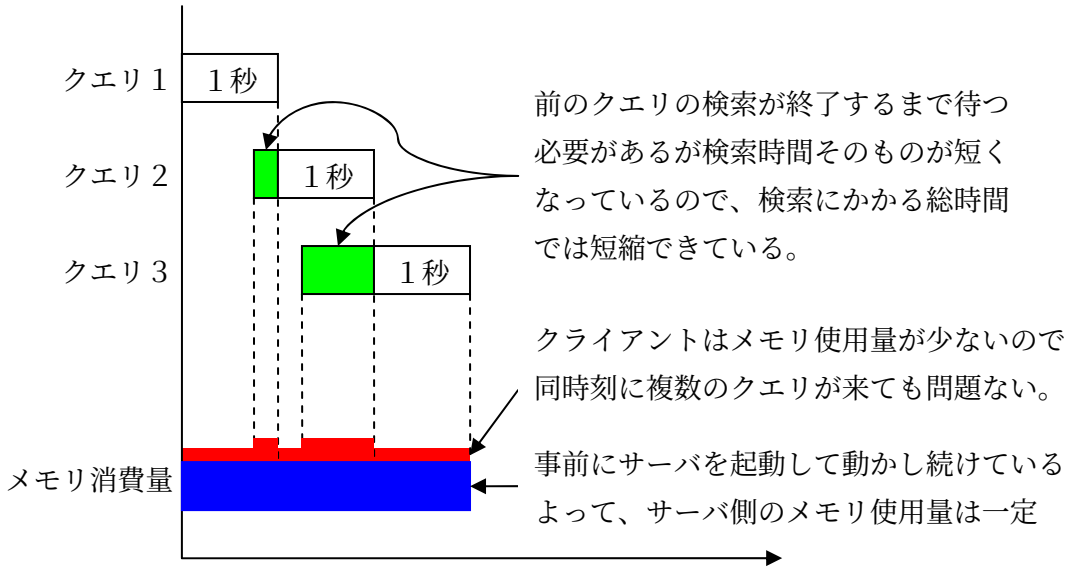


図 150 改善後の検索時間およびメモリ消費量

第12.2.3項　　複合部分構造検索システム

構造活性相関における部分構造検索の利用目的の一つとして、クエリとして入力した化学構造が何かしらの活性を持っていて同じ活性を持つ化合物を探索することを目的とすることがある。ここで、「複数のクエリ構造を入力として用い、全ての部分構造を同時に包含する化合物を探索したい」という要望が化学の専門家から得られた（図 151)。この要求に対して既存の部分構造検索アルゴリズムをそのまま用い、「各クエリを個別に用いて部分構造検索を実行し、全てのクエリに対してヒットの得られた化合物を出力する。」というアプローチを取った場合、図 152 のように複数のクエリ構造が化学構造上で重複しているような化合物も検索結果に含まれてしまう。さらに、図 153 のように一つの原子を介して環が結合しているような構造はスピロ結合と呼ばれる。スピロ結合の様に原子のみで環が結合しているケースでは各々の環の構造活性が存在すると化学の専門家は考えている。また、この様な複合部分構造検索は質量分析装置から得られた複数の MS/MS フラグメントパターンから元の構造を推定するのにも役立つ。この場合は、スピロ結合のケースとは異なり、部分構造間における原子の重複も許したくはない。まとめると、複合部分構造検索には以下の 3 パターンが存在する（図 154)。

- node と edge 両方の重複を許す。
- node の重複のみを許す。
- node も edge も重複を許さない。

クエリ構造間の重複を許す複合部分構造検索は、1 つ目の構造に対する検索結果を、パイプを用いて 2 つ目の部分構造検索の標準入力に用いることで実現できる（図 155)。クエリ構造間の重複を許さない複合部分構造検索を行う為には、一度のバックトラッキング・アルゴリズムの中で複数の化合物を探索するようにすれば対応可能である。ここで、バックトラッキング・アルゴリズムでは、通過した要素に通過フラグを立てながら探索を行っている。node の重複のみを許す複合部分構造検索を行う為には、入力された化学構造の数だけ各 node が通過フラグを持つようにすれば、問題が解決できる。クエリ構造間の重複を許さない複合部分構造検索は、複数の Molfile を引数として実行する仕様とした（図 156)。

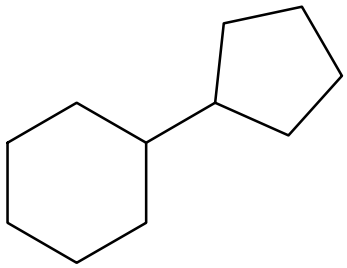


図 151 五員環と六員環を別々の場所で持っている化合物
この様な化合物は五員環と六員環の構造活性を持っていると考えられている

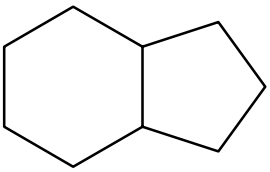


図 152 2 つの環が結合を共有しているケース
この様な化合物は五員環と六員環の構造活性を持っているとは言わないので検索対象から除外したい。

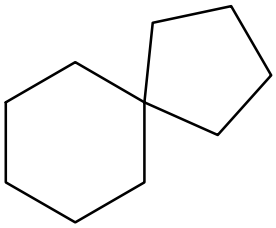
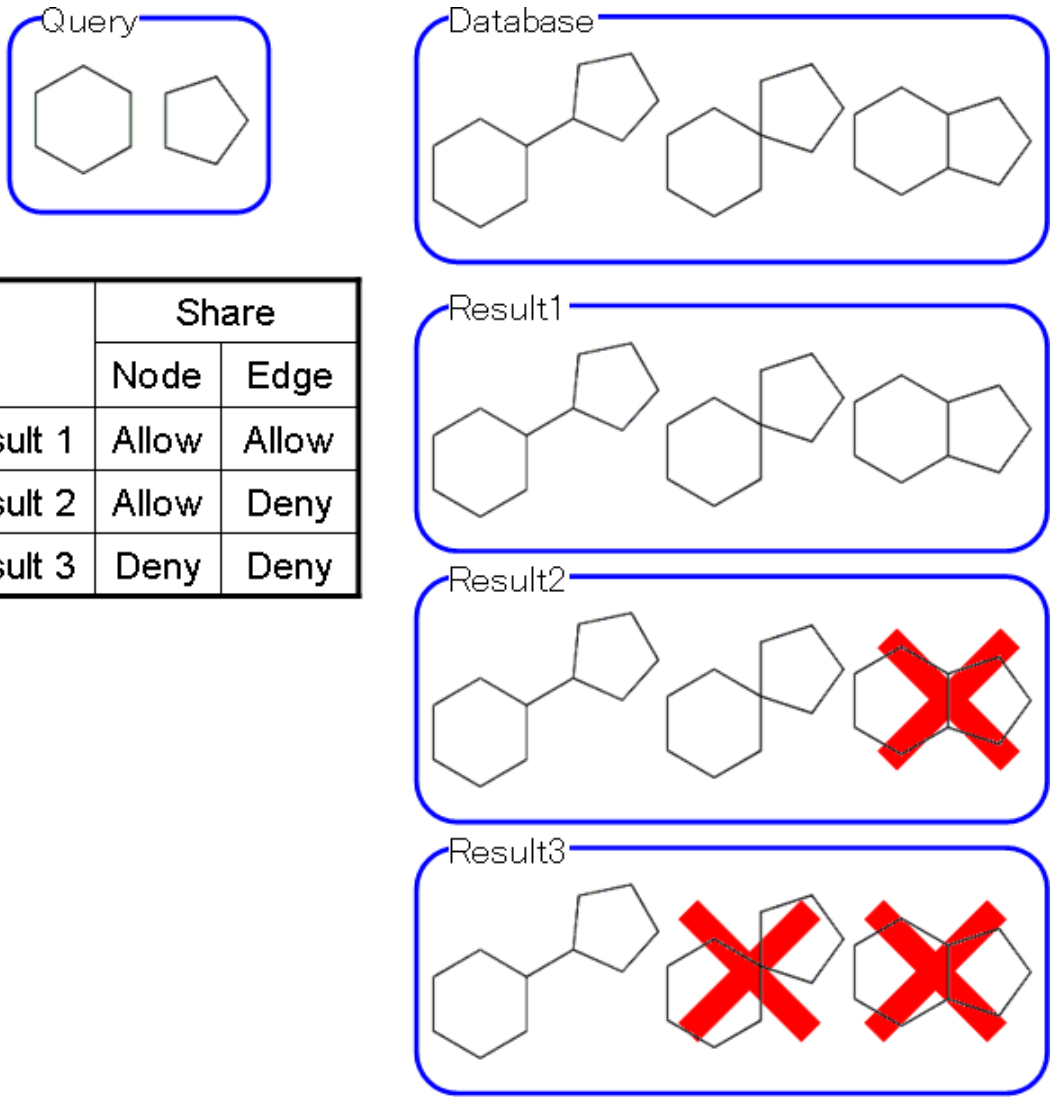


図 153 2 つの環が 1 つの原子を共有しているケース（スピロ結合）
この様な化合物は五員環と六員環の構造活性を持っていると考えられている



	Share	
	Node	Edge
Result 1	Allow	Allow
Result 2	Allow	Deny
Result 3	Deny	Deny

図 154 クエリ構造間の node および edge の共有を許す場合と許さない場合の結果

図中 Database の中央の構造のように 2 つの環が一つの原子を共有する結合をスピロ結合と呼ぶ。五員環と六員環両方の性質を持った化合物の検索は node の重複のみを許した複合部分構造検索を実行することで解答が得られる(Result2)。クエリ構造の五員環と六員環が質量分析装置で得られた MS/MS フラグメントパターンで元の構造を予測したい場合においては、フラグメント構造間の重複を許したくない。この場合はクエリ構造間の node と edge の重複を許さない検索を実行することで解答が得られる(Result3)。

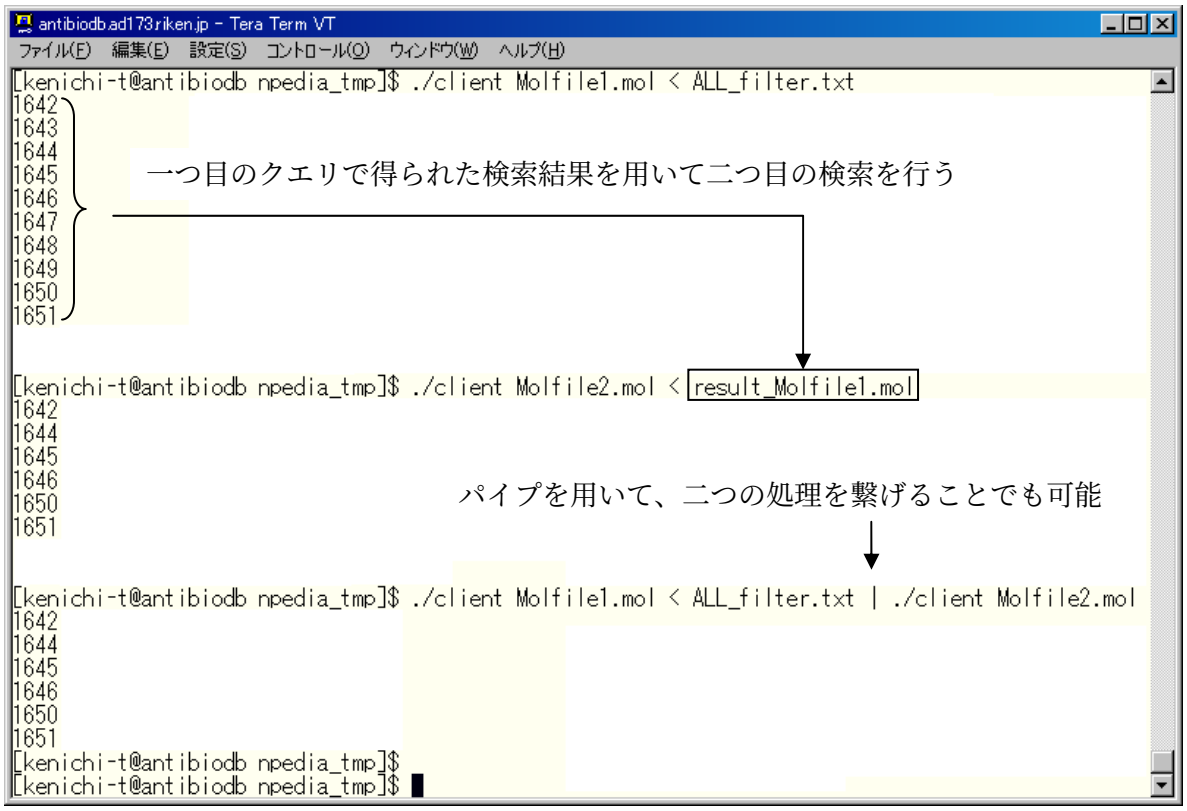


図 155 クエリ構造間の重複を許す場合の実行方法

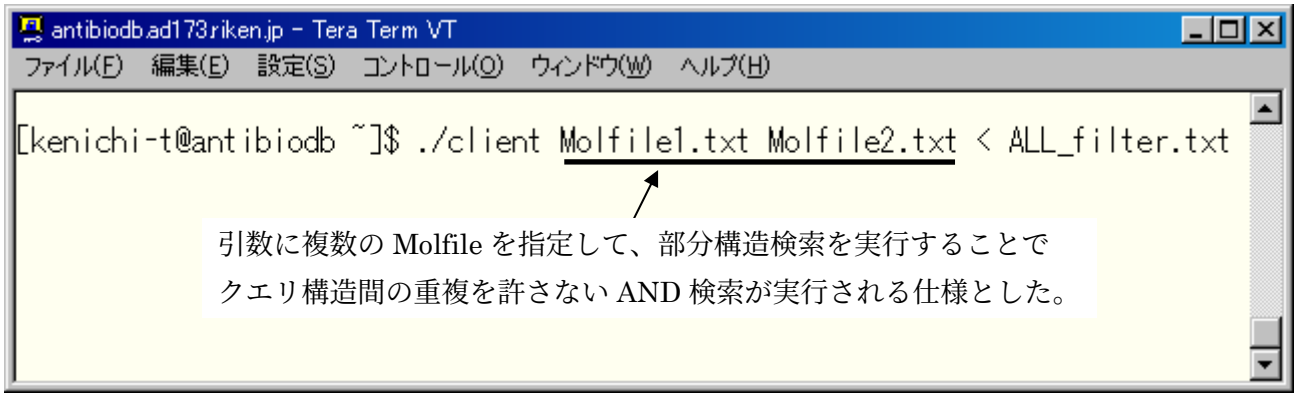


図 156 クエリ構造間の重複を許さない場合の実行方法

第12.3節　結果

本節では、NPEdia および MassBank において、提供したシステムがどのように利用されているのかについて述べる。

第12.3.1項　NPEdia

本研究では、部分構造検索エンジンのみを提供し、WEB 画面の制作等は別の研究者の方が行っている。NPEdia では、複数の化学構造が入力可能なインターフェースを備えている（図 157)。

「draw」ボタンを押すと、JME Molecular Editor[http://www.molinspiration.com/jme/]を用いた化学構造描画画面が立ち上がる（図 158)。描画が完了した後「Submit Molecule」を押すことで、検索ページに反映される（図 159)。重複を許さない検索を行う場合は「Disable edge sharing」にチェックを入れて検索を実行する。

部分構造検索エンジンの実装から 2 年程経過しているが順調に稼働を続けている。

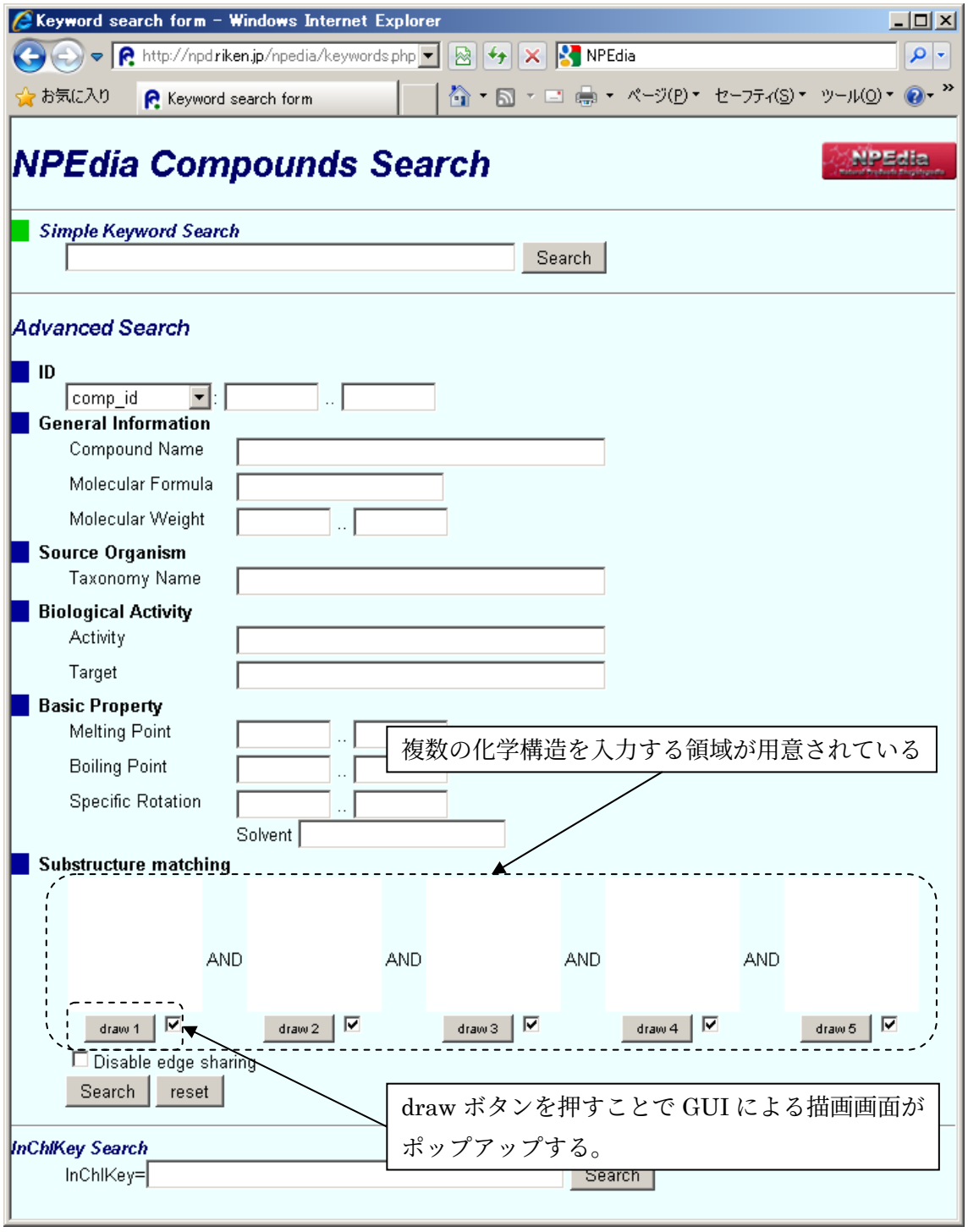


図 157 NPEdia 化合物検索ページ

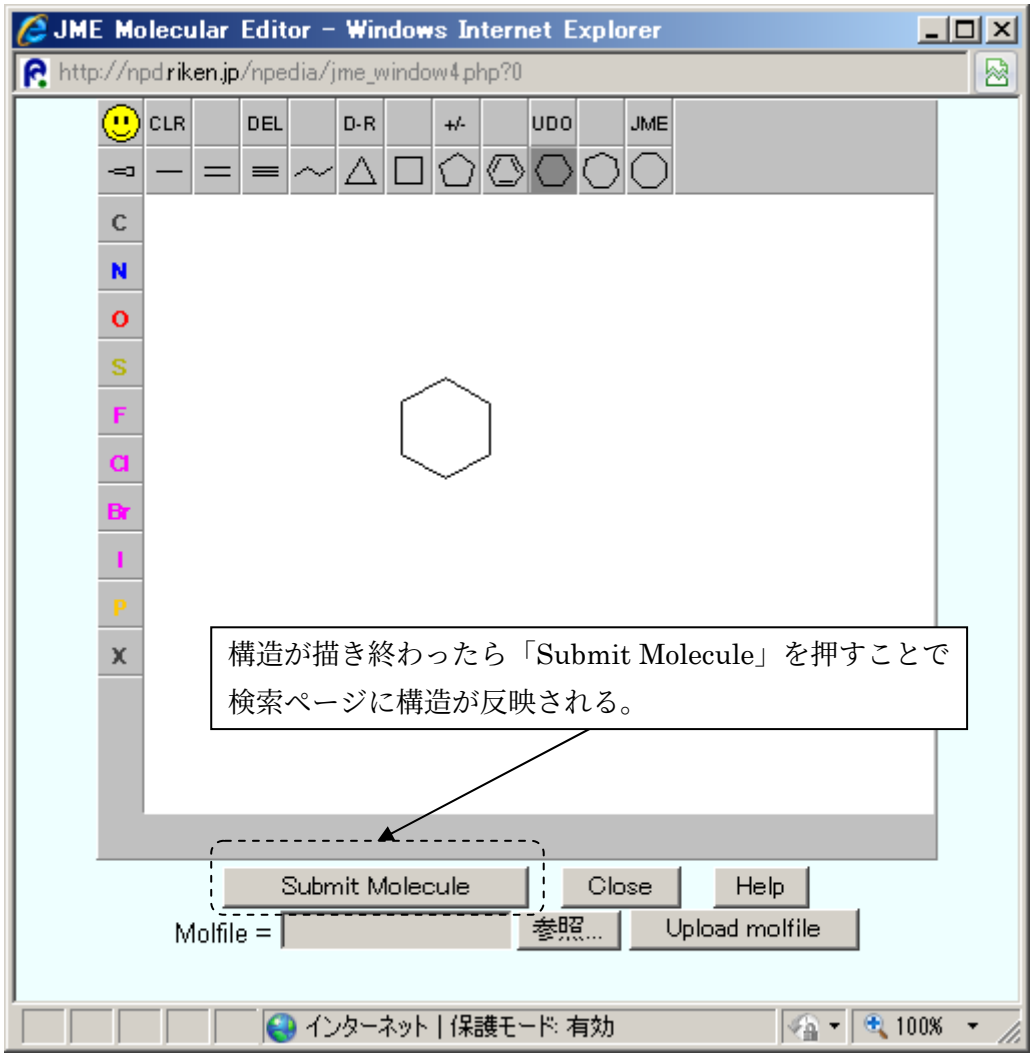


図 158 クエリ構造の入力

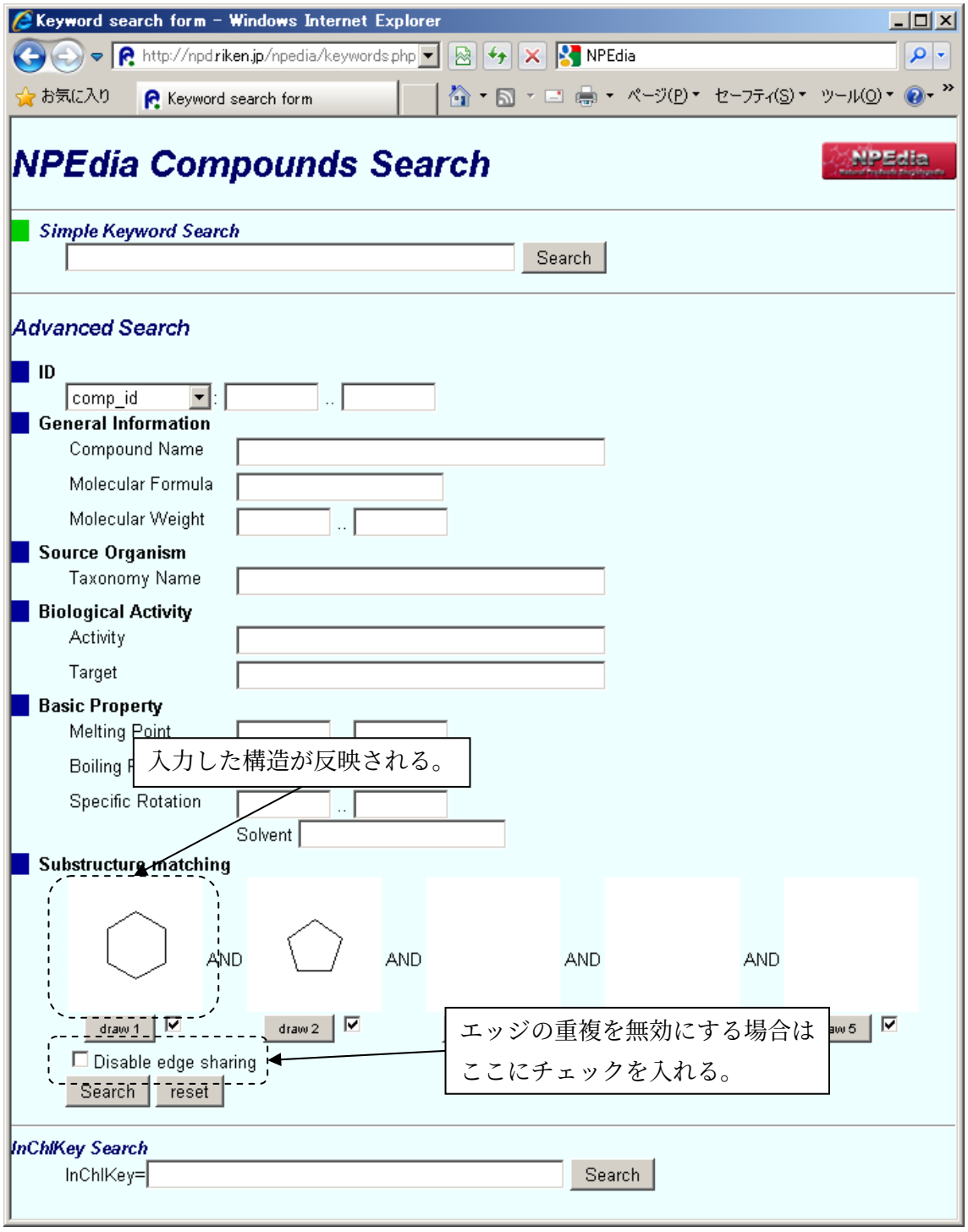


図 159 クエリ構造が反映された化学構造検索ページ

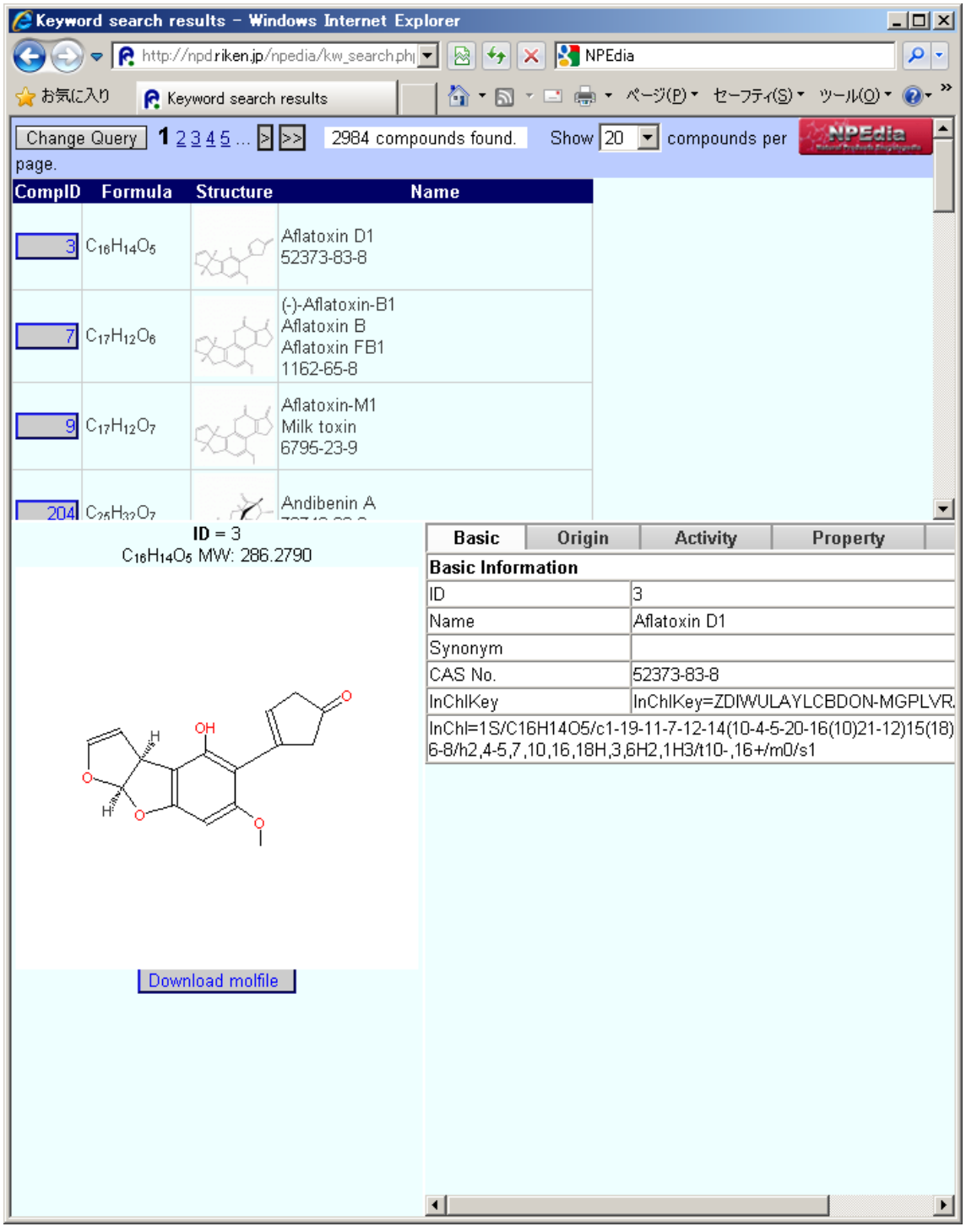


図 160 検索結果

六員環と五員環を持った化合物が検索結果で得られている。

第12.3.2項　　MassBank

MassBank への実装も NPEdia と同様に行った。部分構造検索ページ（図 161）の Edit ボタンを押すことで、JME Molecular Editor を用いた化学構造描画画面が立ち上がる。部分構造検索ページに描画内容を反映させた後に「Search」を押すことで、検索が実行される（図 162）。

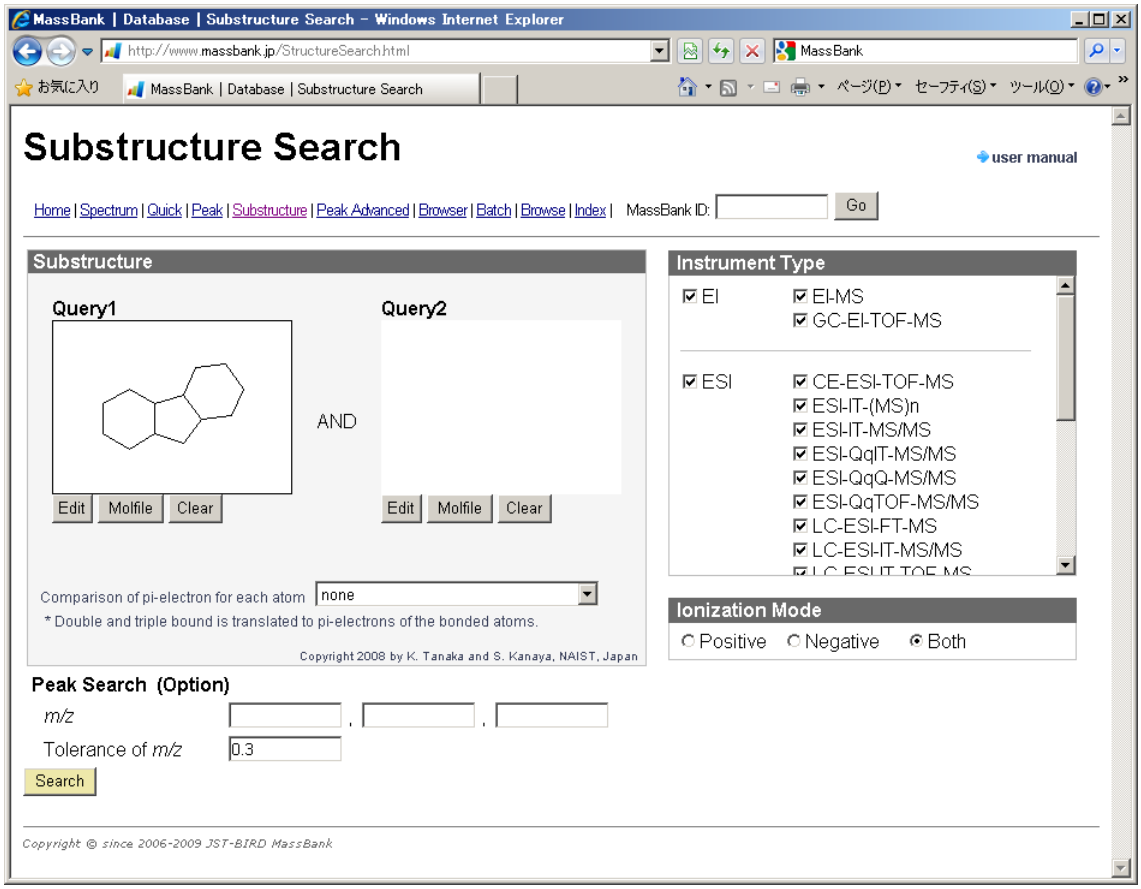


図 161 MassBank 部分構造検索ページ

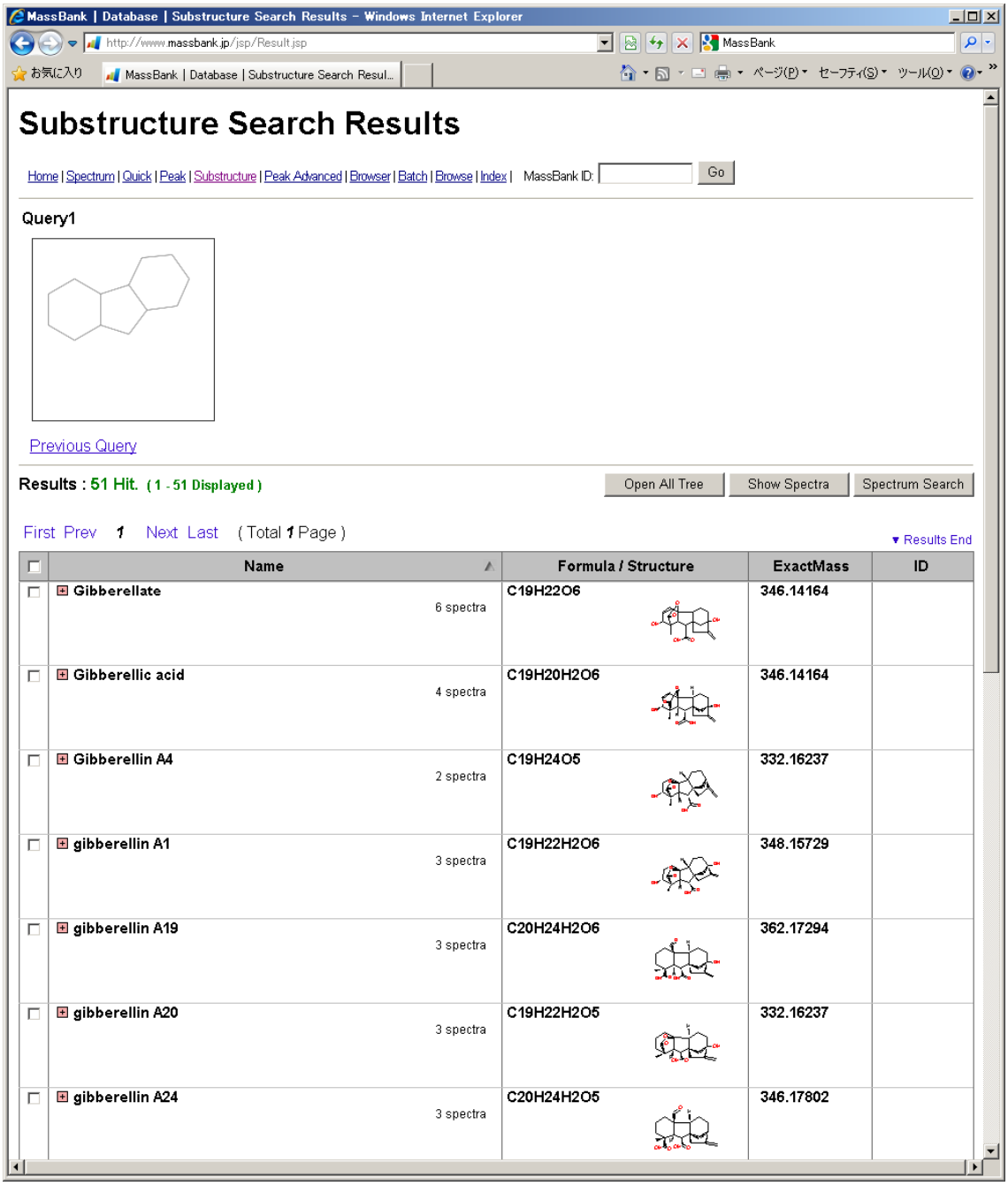


図 162 部分構造検索結果

まとめ

本研究では、現在大部分が未解明な代謝経路を解明する為、部分グラフの包含関係に着目した代謝経路予測手法の提案を行った。提案手法を用いることにより、KNApSAcK に含まれる全 34,653 代謝物に対して経路予測を行っても、2 週間で予測が可能となった。先行研究では NP 困難なアプローチを用いていた為、15,050 代謝物間の経路予測に対して、計算が完了出来たのが 3.34%であった点を踏まえると大幅な進歩であると言える。また、代謝経路を視覚化するアルゴリズムとして Network Self-Organization を提案した。様々な場面で利用され有効性が確認されている自己組織化マップのアルゴリズムを応用することで、与えられた描画領域を無駄なく使って代謝経路を配置することが可能になった。これらのシステムを MetClassifier として実装し WEB 上での公開を行い、誰でも自由に使える状態にした。さらに、外部の研究機関との共同開発において WEB 上で動作する部分構造検索エンジンを開発し、天然化合物データベース NPEdia および高分解能マススペクトルデータベース MassBank に実装を行った。実装の過程で化学の専門家との議論を重ね、複合部分構造検索機能も実装した。これらのデータベースは多くの研究者にとって非常に重要なデータベースであり、部分構造検索エンジンを提供できたことは多くの研究の発展に貢献できたと考えられる。

本研究で開発したシステムが多くの研究者によって利用され、癌研究や食糧問題など現在解決されていない社会問題の解決に少しでも役立てれば幸いである。

謝辞

本論文の審査をして頂きました、奈良先端科学技術大学院大学 情報科学研究科 比較ゲノム学講座 金谷 重彦 教授、システム細胞学講座 小笠原 直毅 教授、比較ゲノム学講座 Md. Altaf-Ul-Amin 准教授、高橋 弘喜 助教に厚く御礼申し上げます。

今日に至るまでの 4 年間御指導を続けてくださった金谷重彦教授には心より御礼申し上げます。奈良先端科学技術大学院大学に来る前の私は、自分の技術を医療や食糧問題の解決に活かしたいという思いだけで、どのような分野が該当するのか右も左も分からない状態でした。4 年間の研究生活を通じて、学内だけではなく色々な方と出会える機会を与えて頂き、それらを通じて今後の研究人生の指針が得られたことは私にとって非常に大きな糧となりました。

Md. Altaf-Ul-Amin 准教授には、特に論文執筆の度に私の拙い英文を何度も校正していただきました。投稿論文をなかなか書き上げることができず精神的にかなり落ち込んでいた時期にとても励ましていただきました。心から感謝いたします。

中村 建介 特任准教授には、日常におけるコミュニケーションの中で様々なお話を聞かせて頂きました。私にとって化学や生物学のような情報科学以外の分野に関する知識に触れる機会は非常に重要でとてもありがたかったです。また、本研究で利用させていただいた KNApSAcK の化学構造データの多くは中村先生の手によって構築されたものです。この様なデータベースの存在の上に私の研究が成り立っているのだと思います、心から感謝いたします。

高橋 弘喜 助教には、4 年間に渡り初めは先輩として最後の 1 年は助教として、様々なアドバイスをして頂きました。バイオサイエンスの実験全般に関する話題から、統計などの情報科学的な話題等様々な方面におけるアドバイスをいただきました。心から感謝いたします。

平井 晶 秘書には、秘書業務だけでなく KNApSAcK に関わる様々なシステム開発も行っておられる状況において、研究生活のあらゆる面でサポートをして頂きました。心から感謝いたします。

和田 眞昌 様には、研究室内のコンピュータに関わるあらゆる業務を通じて、何度も助けて頂きました。研究の終盤で学内が停電になった際、とても心強かったです。本当にありがとうございました。

現在から過去に渡り比較ゲノム学講座に在籍し、KNApSAcK データベースの開発に携わってこられた多くの先輩方、特に中心的役割を果たしていた真保 陽子様には心から感謝いたします。皆様のおかげで本研究を行うことが出来ました。

川端 猛 准教授には常日頃から、「こういうことはちゃんと勉強している？」「こういうことは知っておかなければいけないよ？」と私の研究活動に足りない部分を何度も指摘していただきました、川端先生の厳しい助言はどれも重要なことばかりで、社会に出る前に

厳しく指導していただけたことは私にとって大きな糧になっていると思います。心から感謝申し上げます。

京都大学 化学研究所 バイオインフォマティクスセンター 時松 敏明 特任助教、小寺 正明 特任助教には、これまでに代謝経路予測を行ってきた研究者の視点から、予測した代謝経路に関するアドバイスや過去の先行研究に関する情報等、数多くのアドバイスを頂きました。心から感謝いたします。

理化学研究所 長田抗生物質研究室 長田 裕之 主任研究員、斎藤 臣雄 専任研究員、渡辺 信元 専任研究員、今野 英明 協力技術員には本研究において様々な助言をいただきました。先にも述べましたが私の目的は自分の技術を医療や食糧問題の解決に活かすことで、その為に研究者を目指しています。長田抗生物質研究室で行われている研究は私の目的としていた社会問題を解決しようという研究そのもので、この様な環境で研究する機会を与えてくださった長田先生には心から感謝しています。理化学研究所に滞在させていただいている間、斎藤先生には連日にわたって数時間におよぶディスカッションをして頂き、化学者が何を重要視していて、情報科学者である私が何をすればいいのか大きな指針を示していただきました。本研究で開発した機能の多くはこれらのディスカッションが元になっています。Maximum Frequent Connected subgraph 抽出機能の開発は、渡辺先生からの「実験で得られた化合物の共通性を見出したい」といった話が開発のスタートになっています。開発の際に貴重な実験データを提供していただきました。また、プロトタイプシステムを提示した際に「こういう情報が見たい」、「こういう機能があったらうれしい」等様々な助言をいただきました。今野様には特に部分構造検索エンジンの NPEdia への実装の際に、システムの仕様決定のアイデア提供をしていただきました。また NPEdia データベース構築に関わる様々な技術に関してご教授いただきました。長田抗生物質研究室の皆様には心から感謝申し上げます。

本研究で開発した MetClassifier には以下のライブラリおよび開発環境を利用させていただきました。この場を借りて感謝申し上げます。

OpenGL	[http://www.opengl.org/]
GLUT	[http://www.opengl.org/resources/libraries/glut/]
AntTweakBar	[http://www.antisphere.com/Wiki/tools:anttweakbar]
Microsoft Visual C++ 2008 Express Edition	[http://www.microsoft.com/japan/msdn/vstudio/Express/]

最後に、以前の大学を辞めてから 8 年間、多くの方に心配をかけました、いつまでも連絡を取り続け、何度も励まし続けてくれた友人の皆様、私を支え続けてくれた家族に心から感謝の意を表します。

付録 A

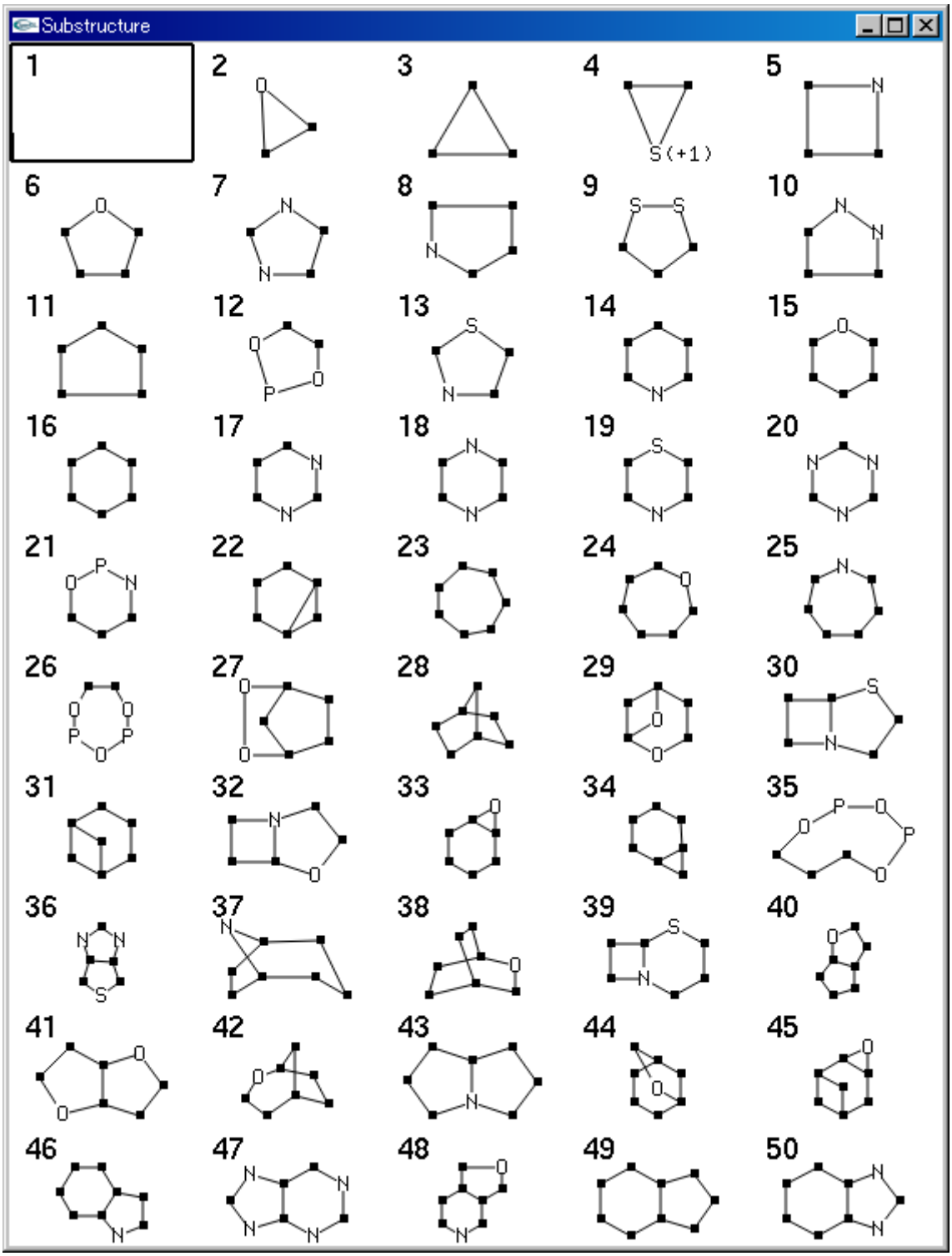


図 163 RPAIR データベース main から抽出した Framework 1

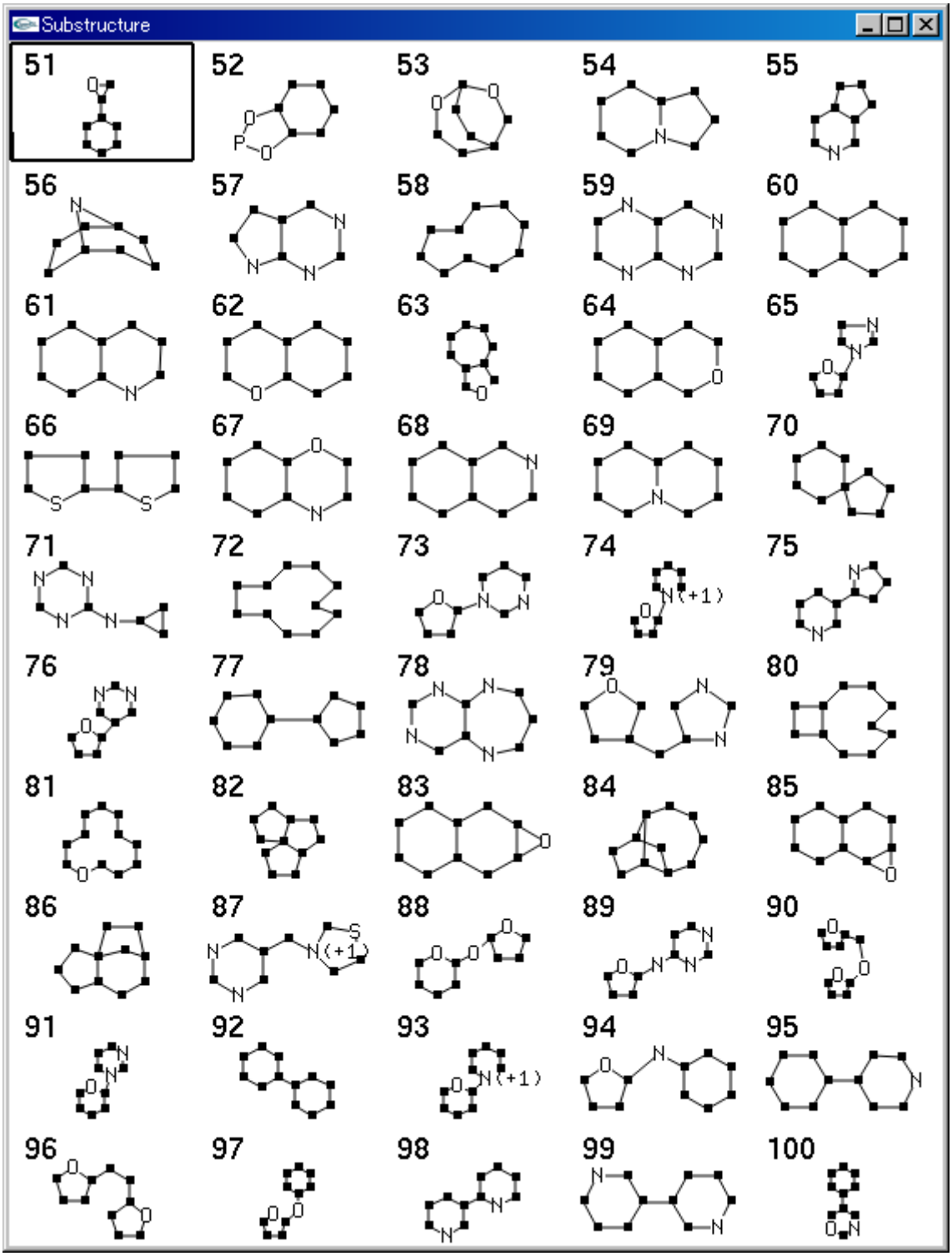


図 164 RPAIR データベース main から抽出した Framework 2

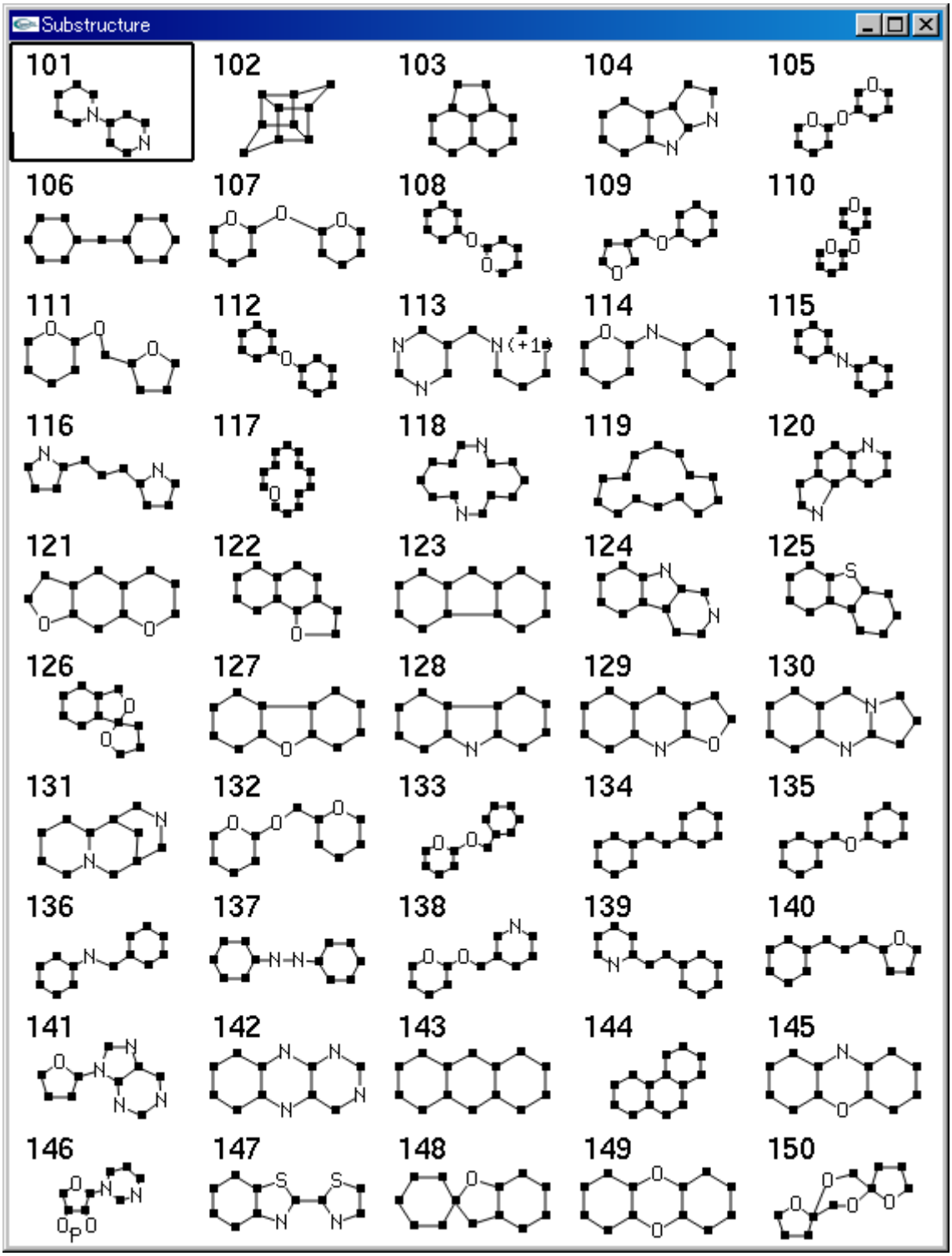


図 165 RPAIR データベース main から抽出した Framework 3

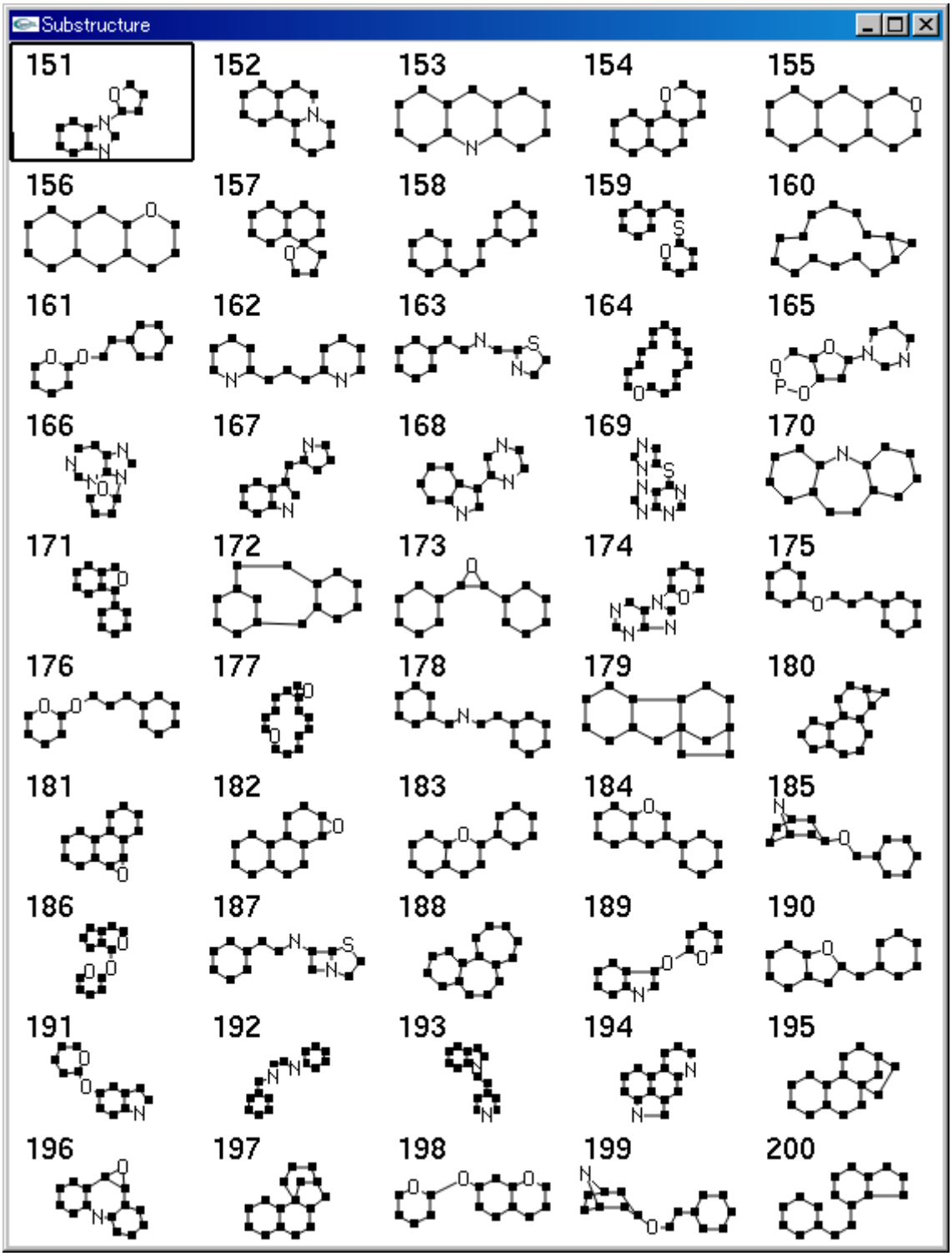


図 166 RPAIR データベース main から抽出した Framework 4

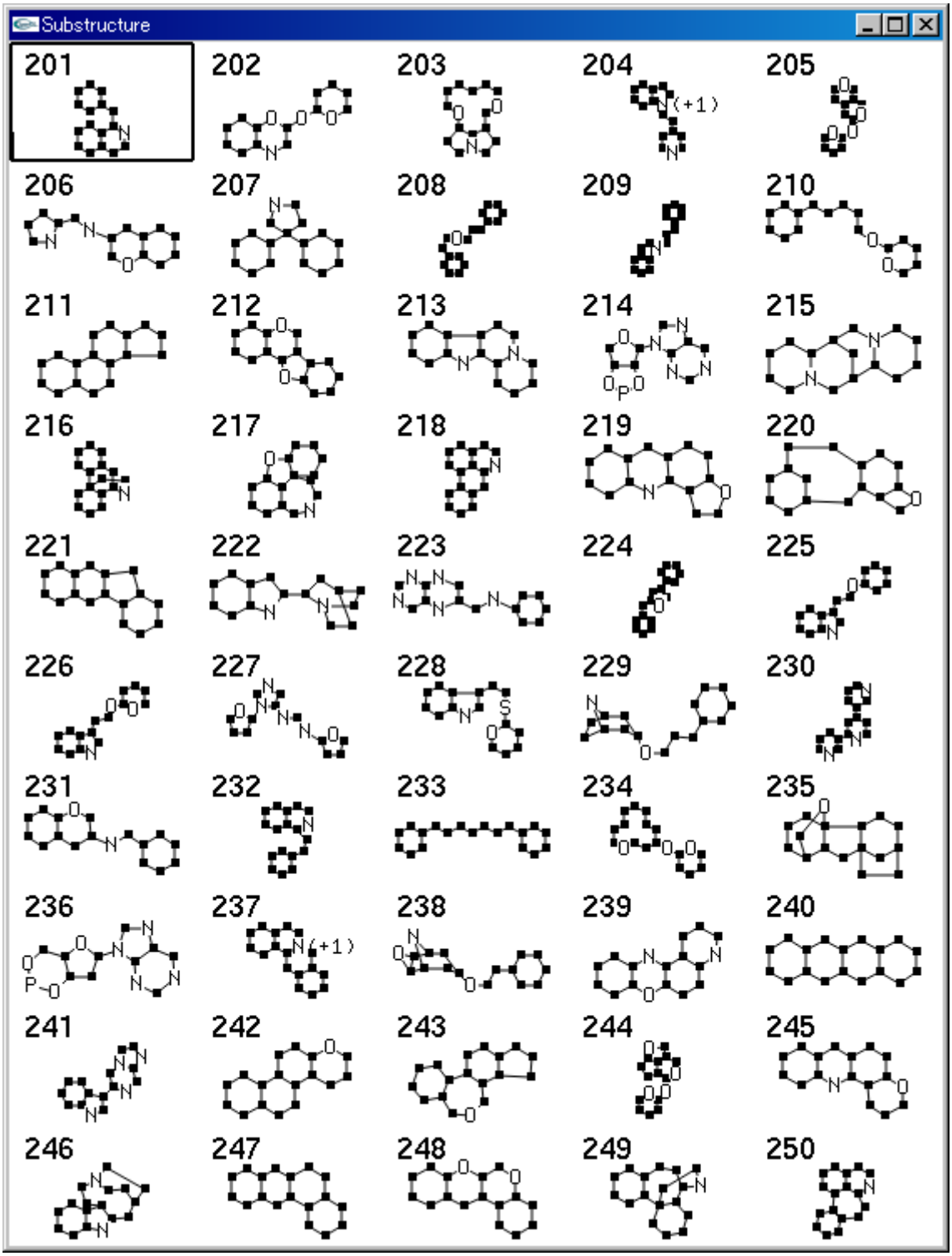


図 167 RPAIR データベース main から抽出した Framework 5

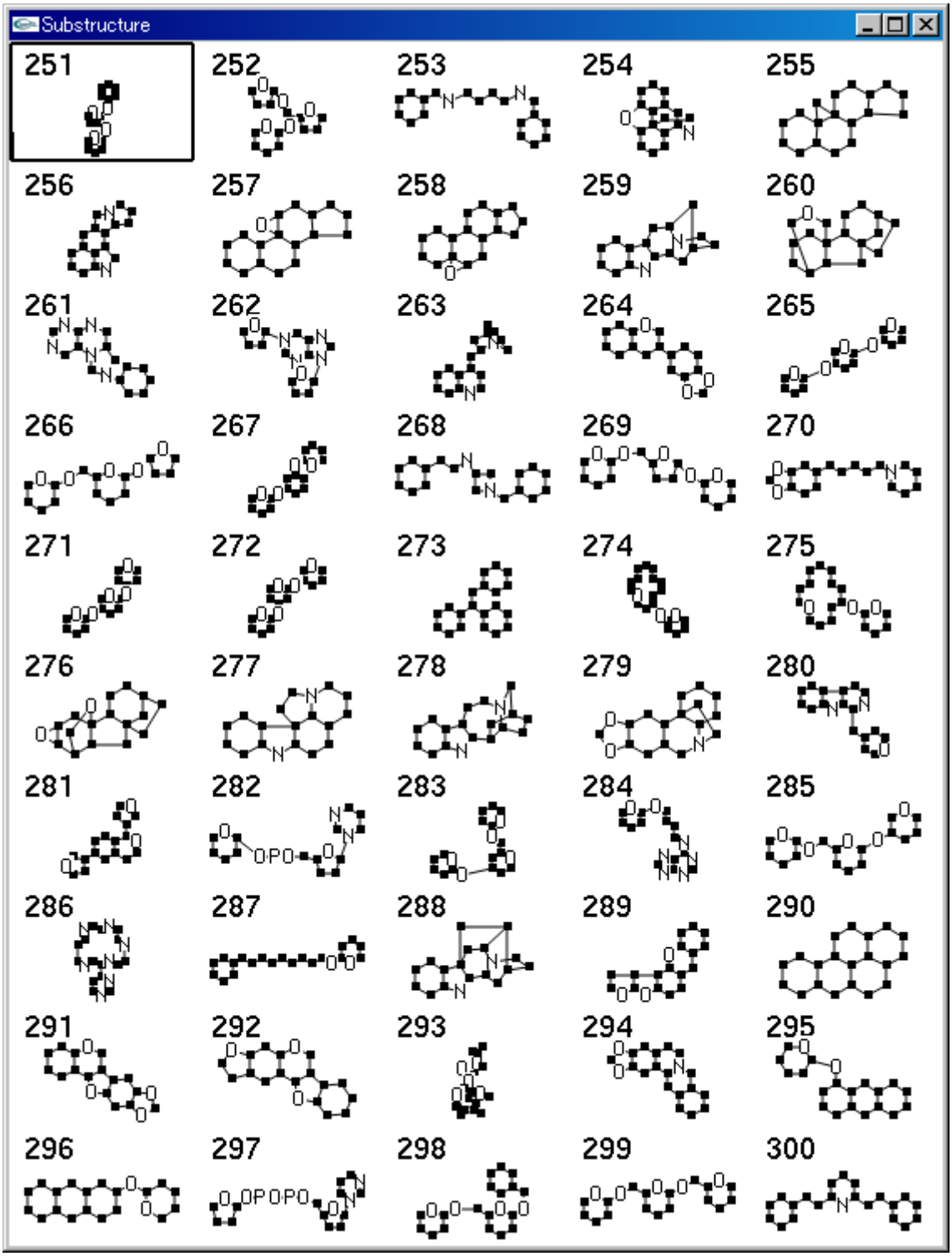


図 168 RPAIR データベース main から抽出した Framework 6

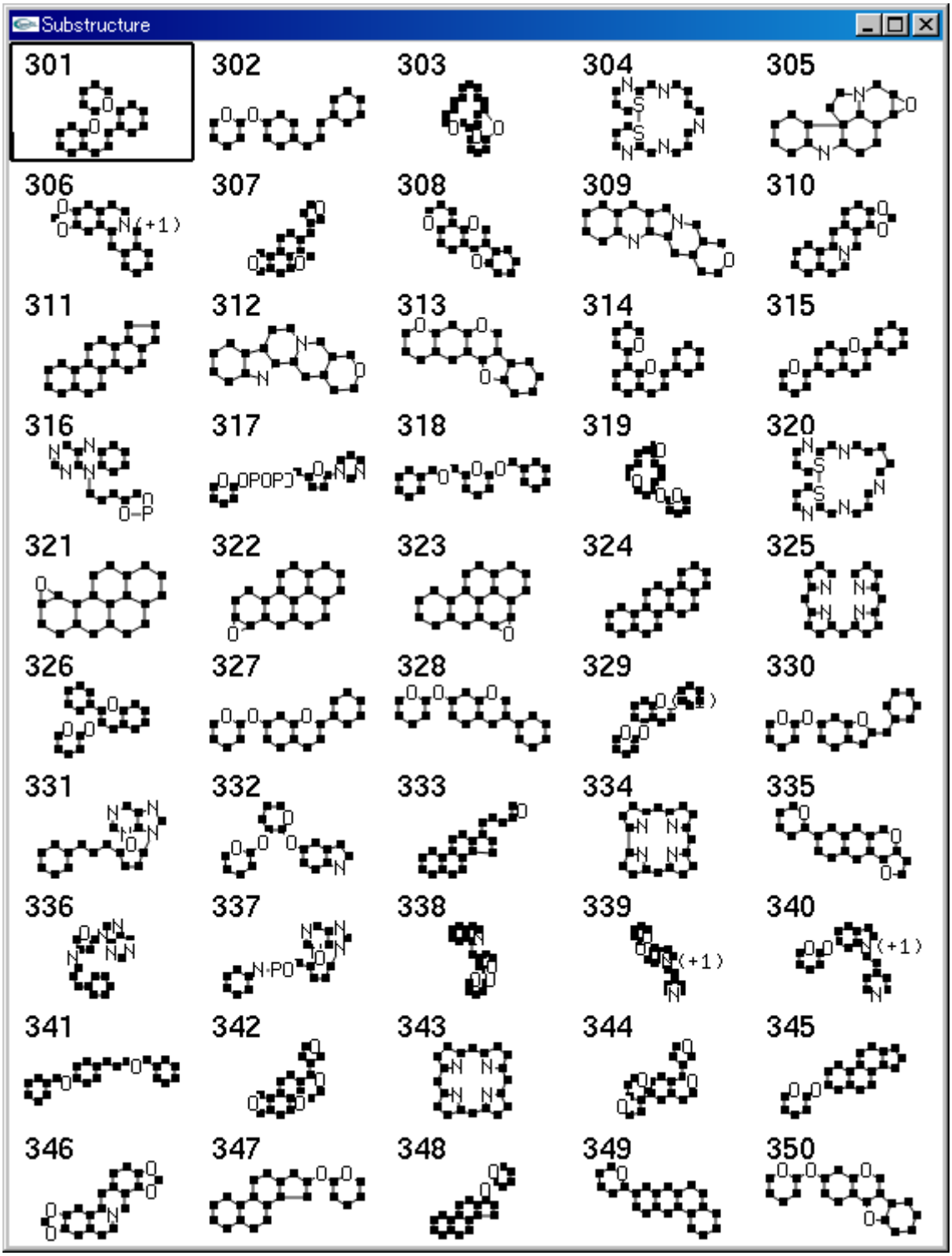


図 169 RPAIR データベース main から抽出した Framework 7

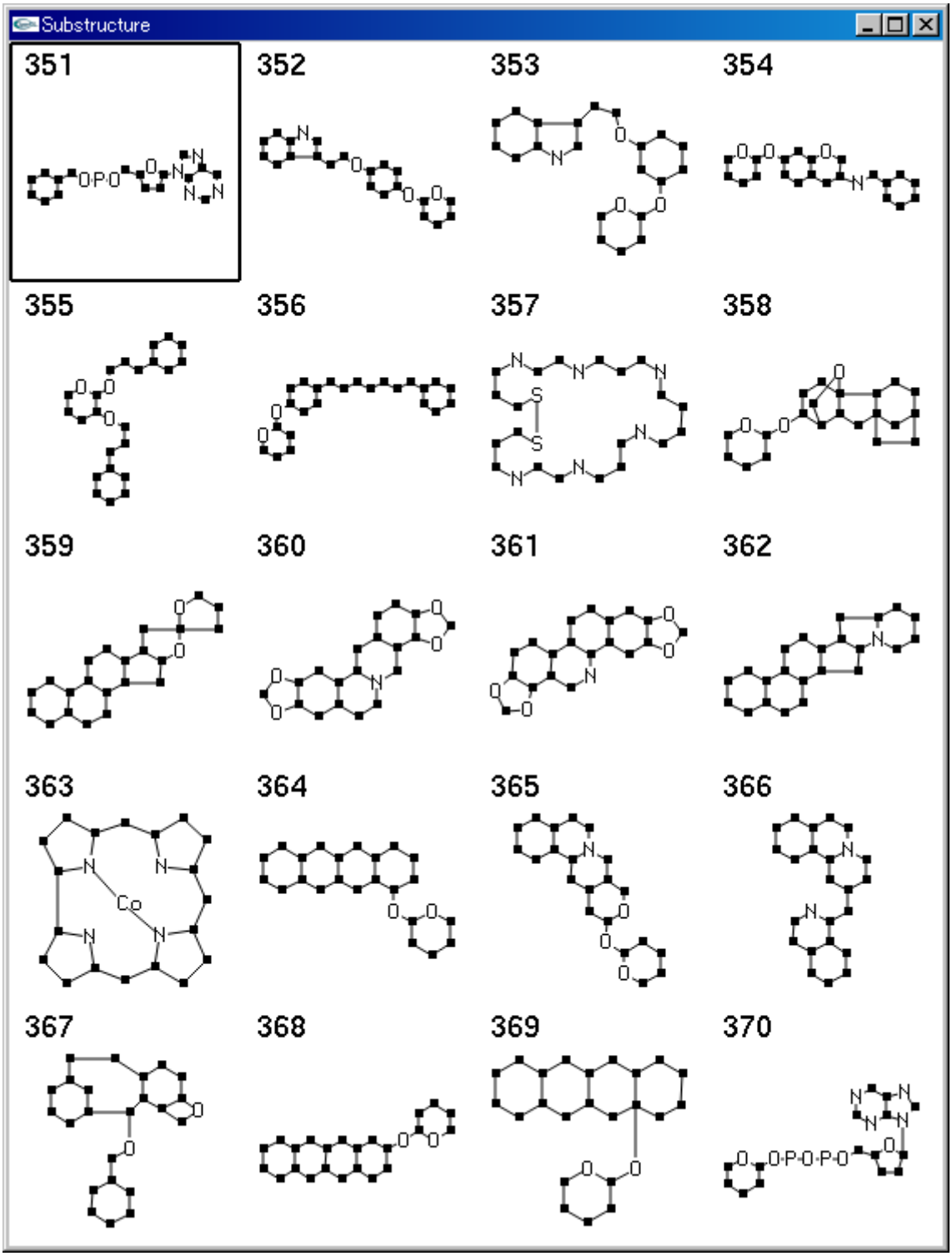


図 170 RPAIR データベース main から抽出した Framework 8

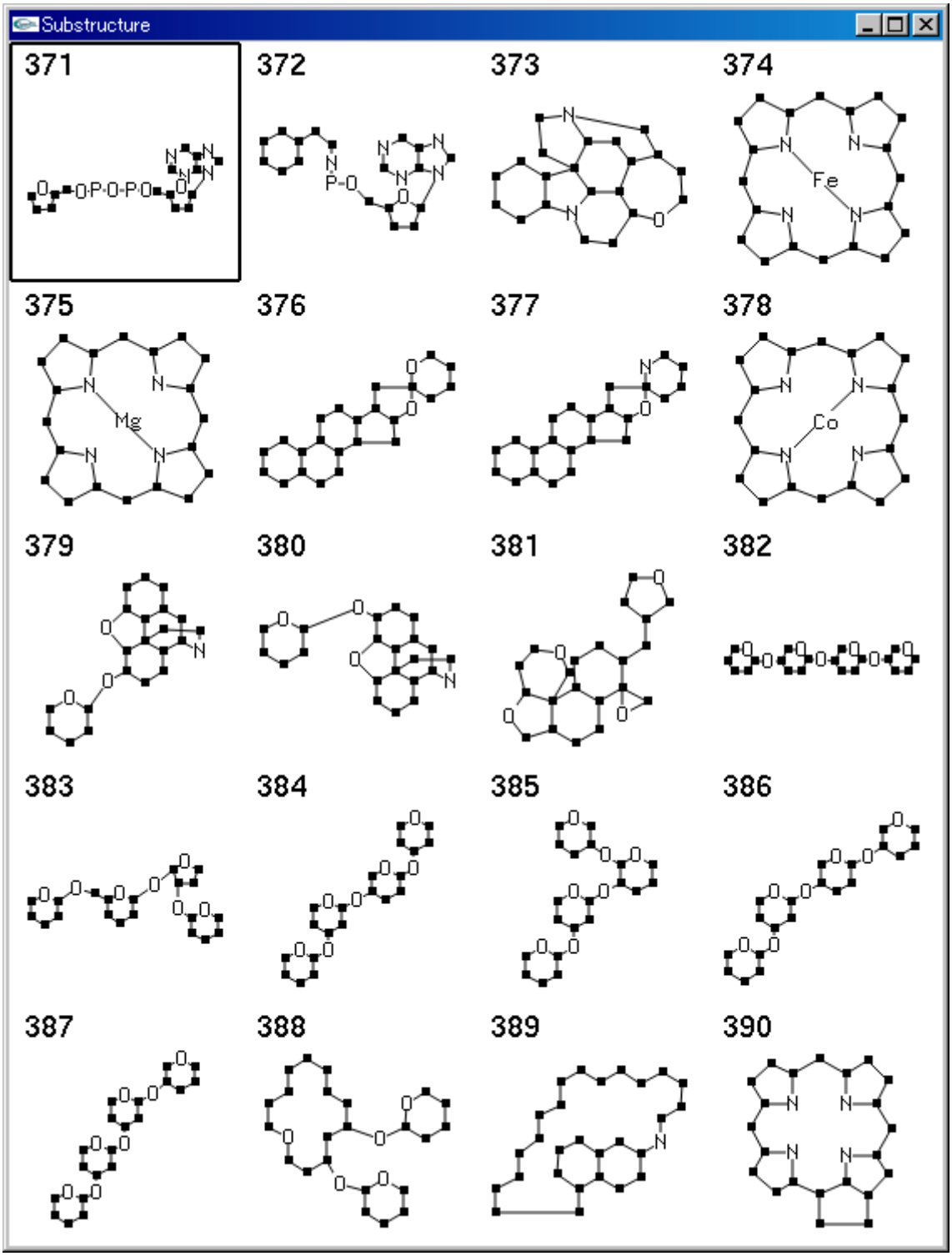


図 171 RPAIR データベース main から抽出した Framework 9

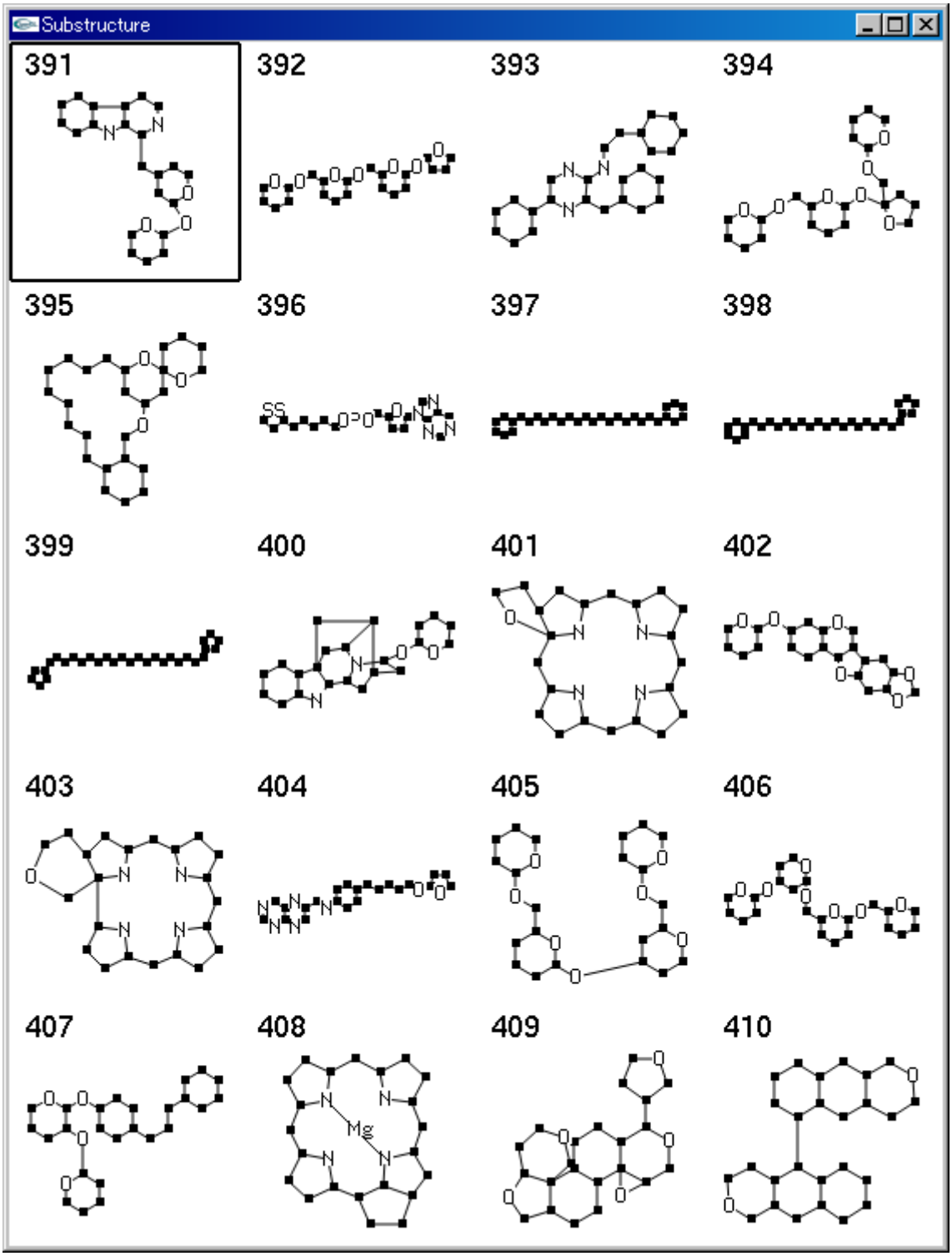


図 172 RPAIR データベース main から抽出した Framework 10

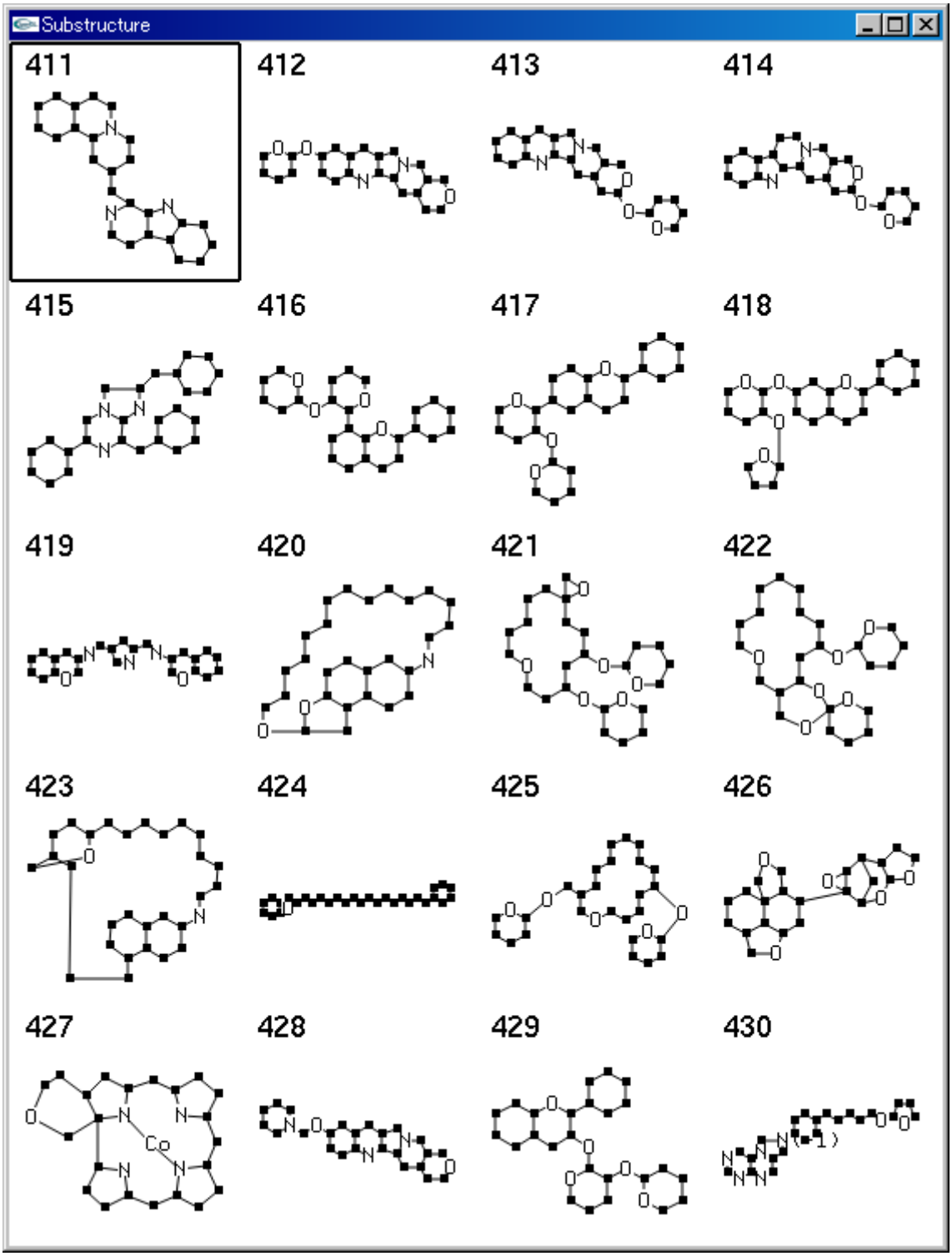


図 173 RPAIR データベース main から抽出した Framework 11

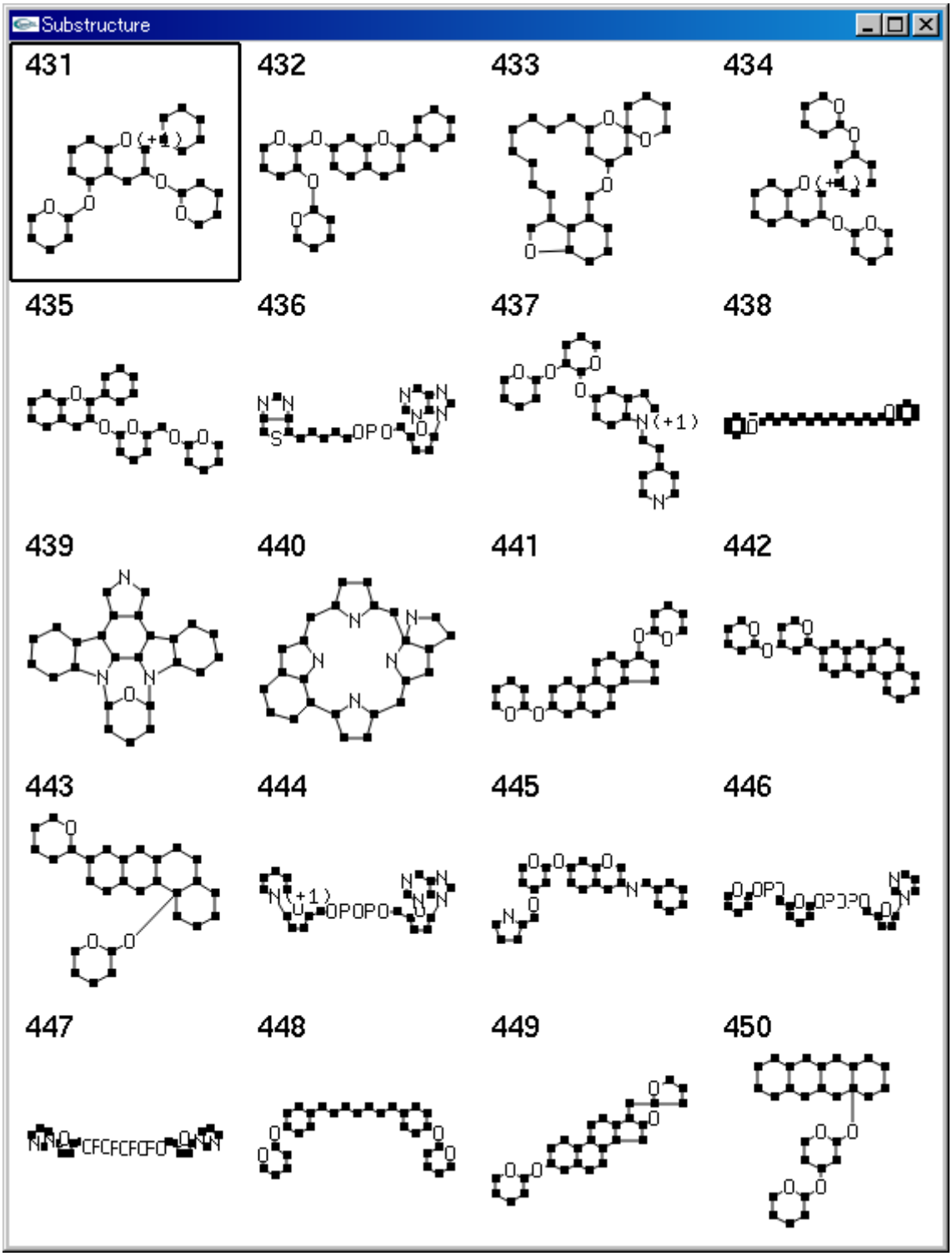


図 174 RPAIR データベース main から抽出した Framework 12

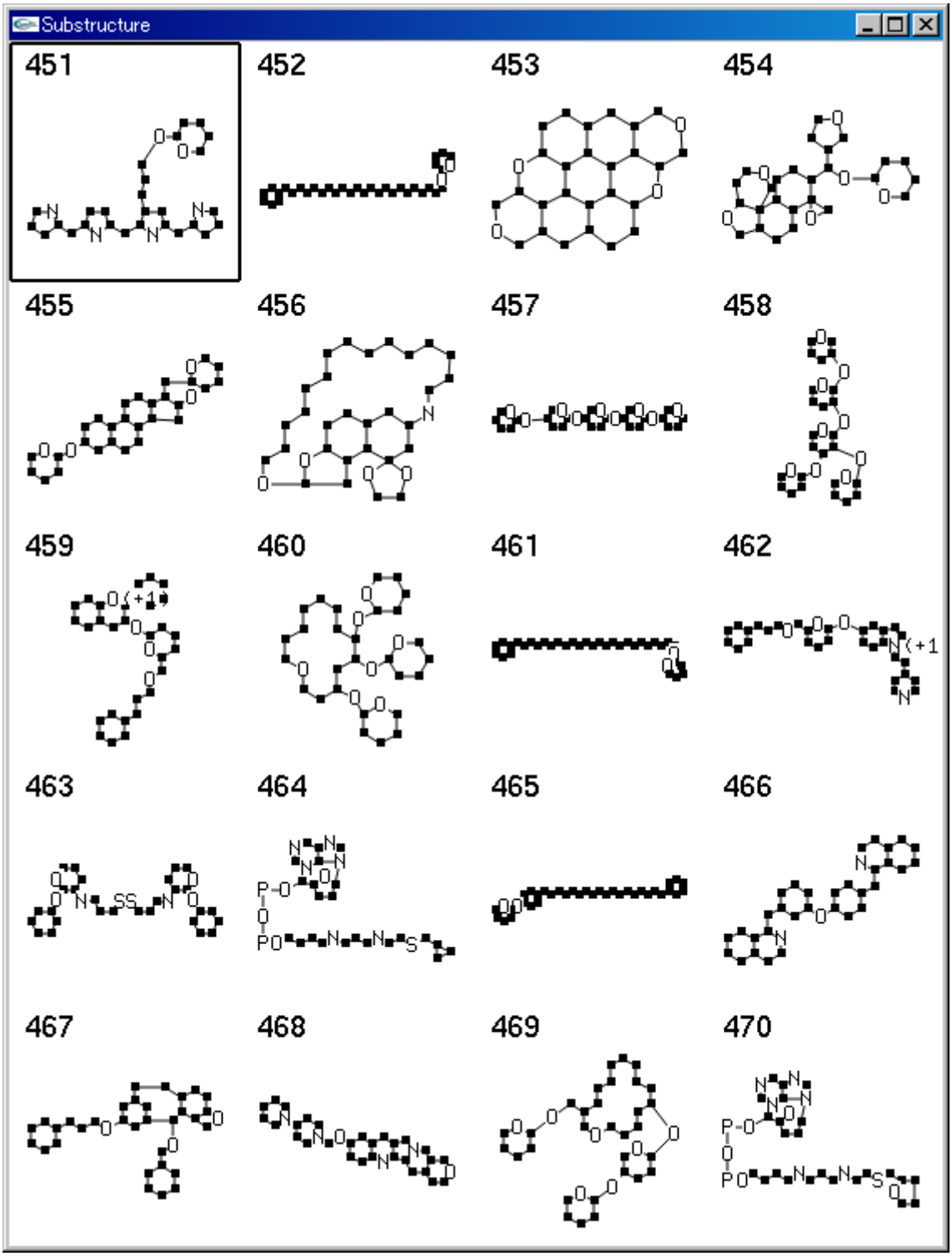


図 175 RPAIR データベース main から抽出した Framework 13

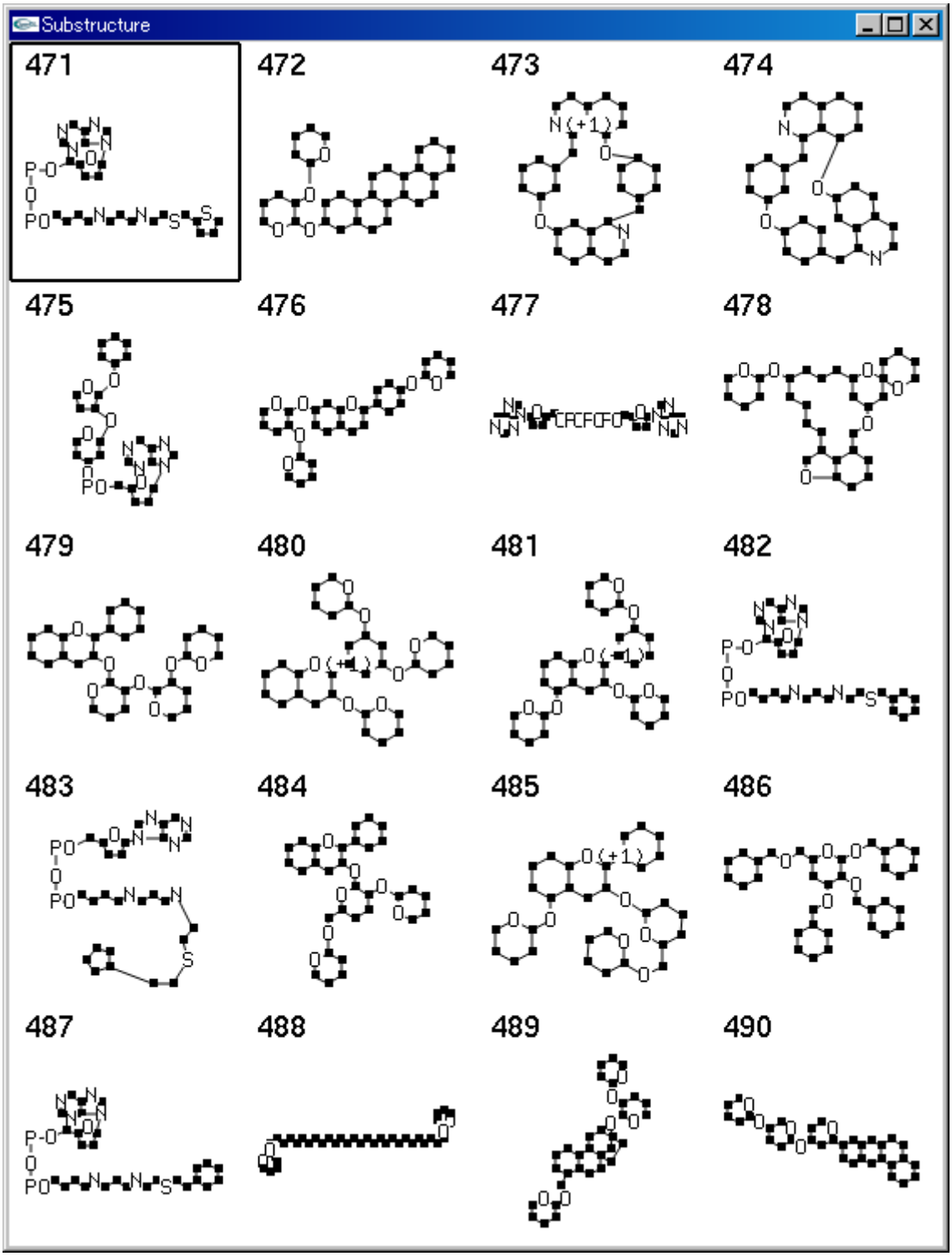


図 176 RPAIR データベース main から抽出した Framework 14

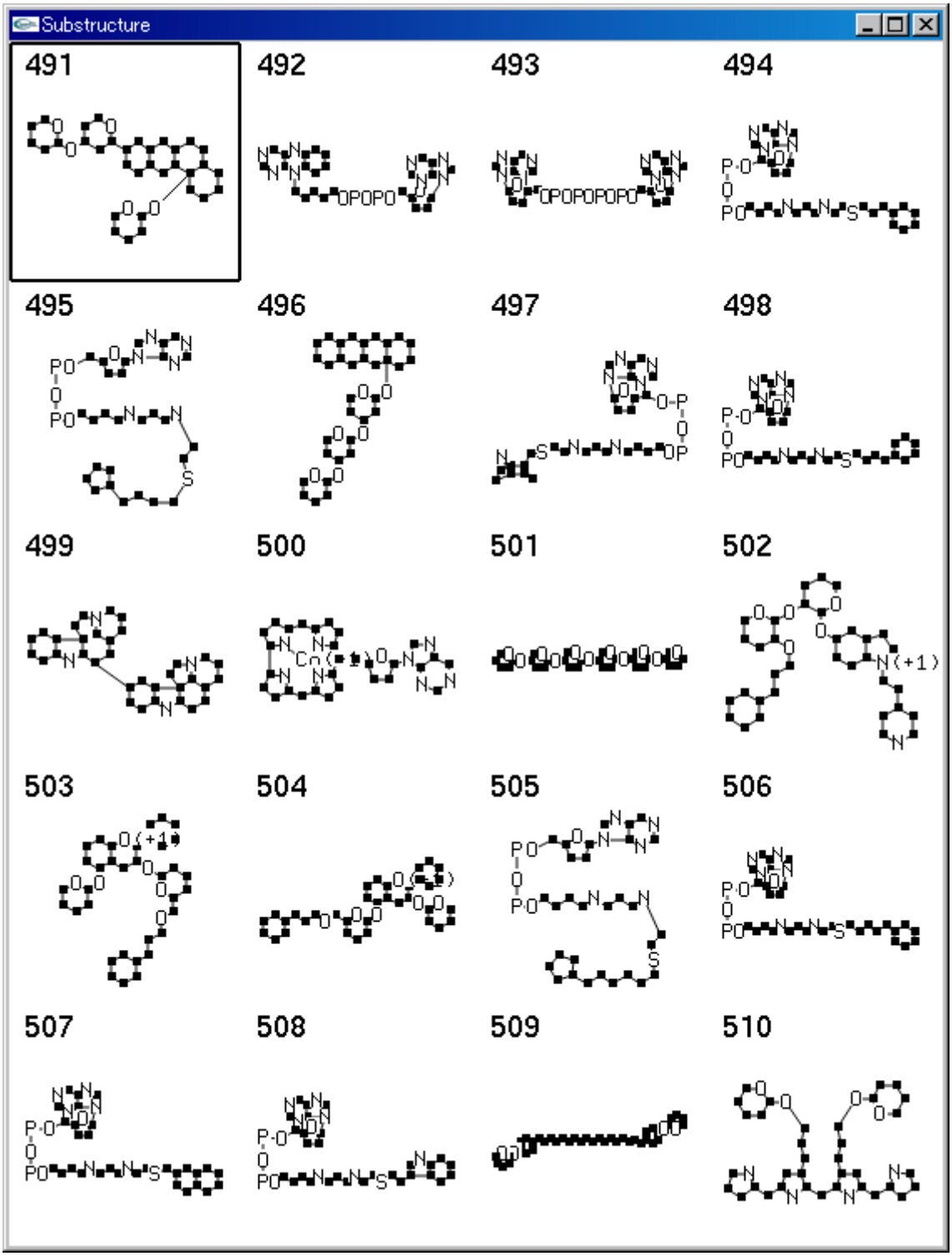


図 177 RPAIR データベース main から抽出した Framework 15

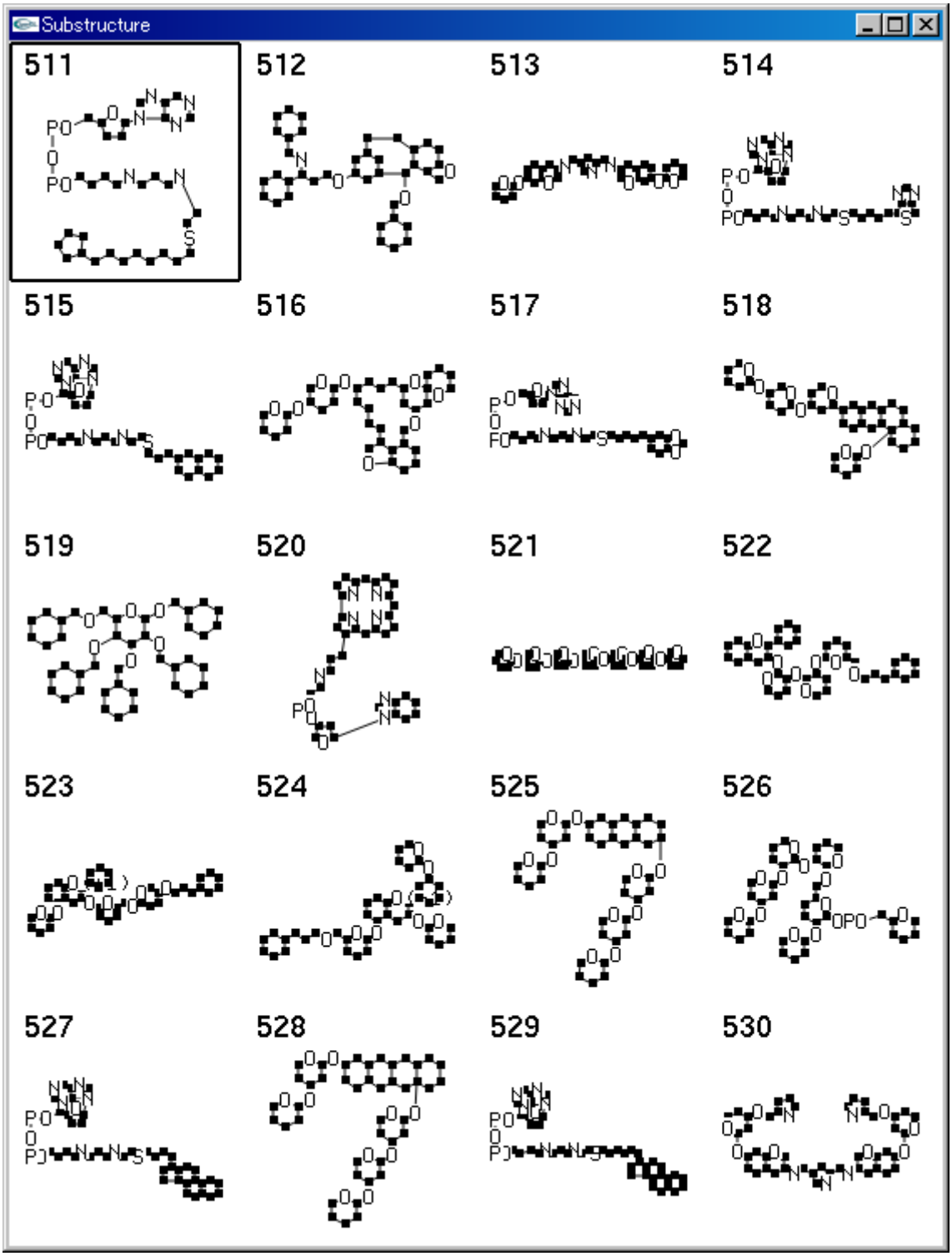


図 178 RPAIR データベース main から抽出した Framework 16

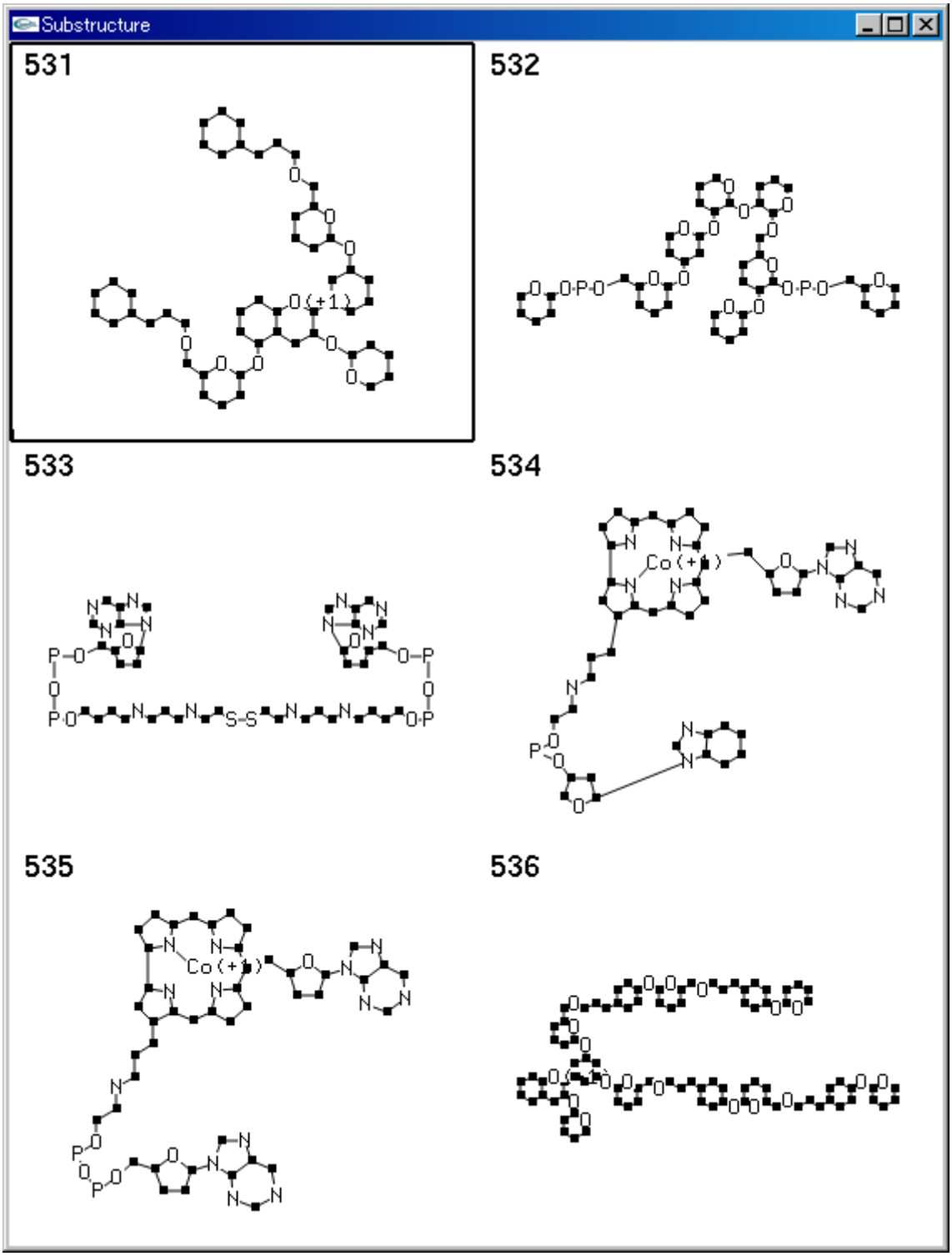
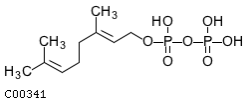
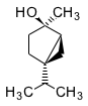


図 179 RPAIR データベース main から抽出した Framework 17

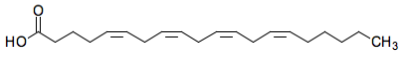
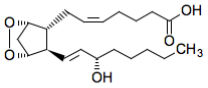
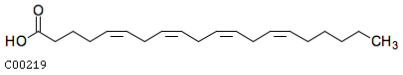
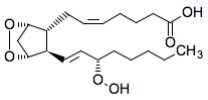
付録 B

KEGG RPAIR main に含まれる代謝物ペアの内、提案手法により Framework レベルの代謝経路予測を行った結果、Framework 間に Ancestore-Descendant の関係が得られた代謝物ペア。各表上部の番号は Framework ID を意味している。表一列目の ID は KEGG RPAIR ID である。

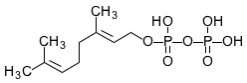
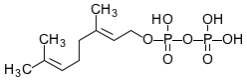
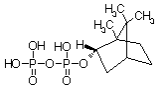
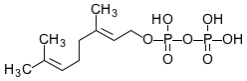
1-22

RP01905		

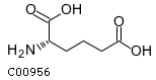
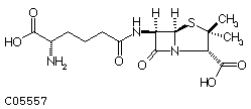
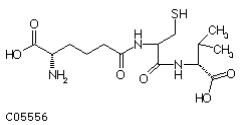
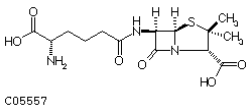
1-27

RP01626		
RP01619		

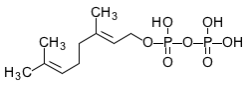
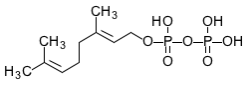
1-28

RP01904		
RP01906		
RP13370		

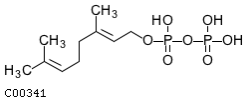
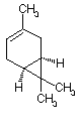
1-30

RP06076		
RP04487		

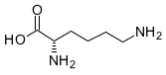
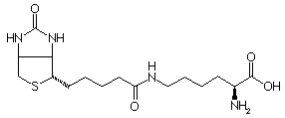
1-31

RP05354	 C00341	 C06307
RP05353	 C00341	 C06308

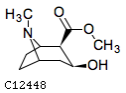
1-34

RP12961	 C00341	 C09839

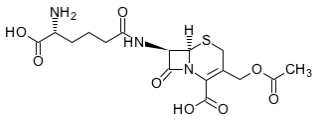
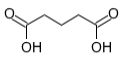
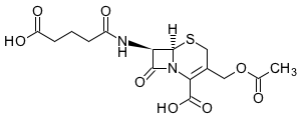
1-36

RP01285	 C00047	 C05552

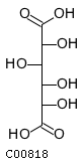
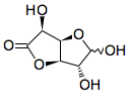
1-37

RP09164	C00132	 C12448

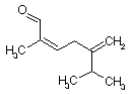
1-39

RP02733	 C00033	 C00916
RP11157	 C00489	 C15666

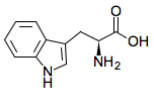
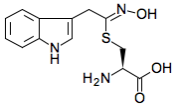
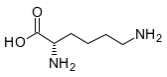
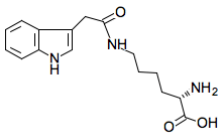
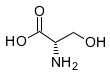
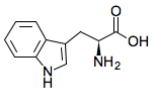
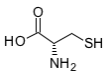
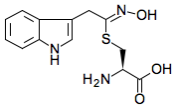
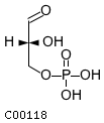
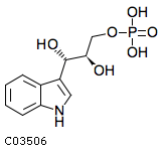
1-41

RP02637	 C00818	 C02670

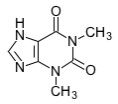
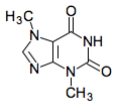
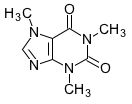
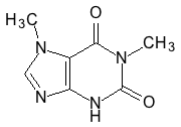
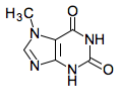
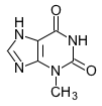
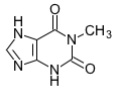
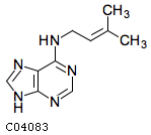
1-45

RP03667	 C04435	 C02759

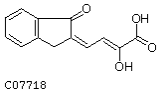
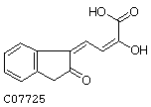
1-46

RP01046	 C00022	 C00078
RP10374	 C00022	 C16518
RP02760	 C00047	 C04211
RP00587	 C00065	 C00078
RP13280	 C00097	 C16518
RP02139	 C00118	 C03506

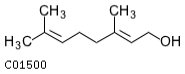
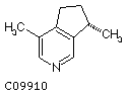
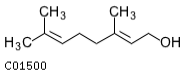
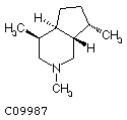
1-47

RP12268	 C00067	 C07130
RP12269	 C00067	 C07480
RP12272	 C00067	 C07481
RP12270	 C00067	 C13747
RP10864	 C00067	 C16353
RP10865	 C00067	 C16357
RP10866	 C00067	 C16358
RP05307	 C07330	 C04083

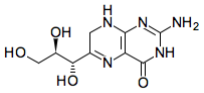
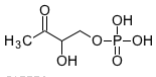
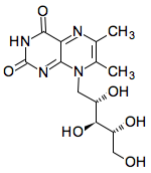
1-49

RP05122	 C00022	 C07718
RP05128	 C00022	 C07725

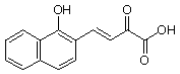
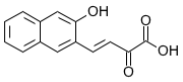
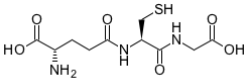
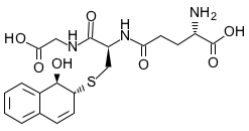
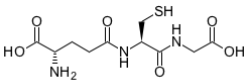
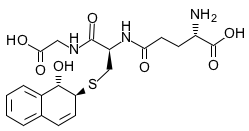
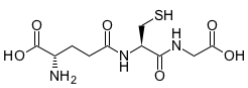
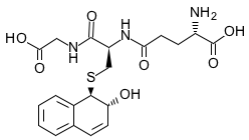
1-55

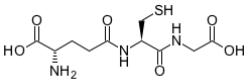
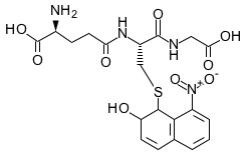
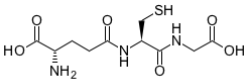
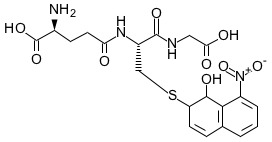
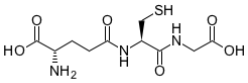
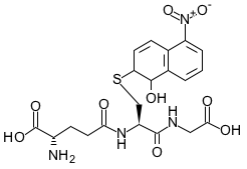
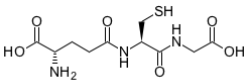
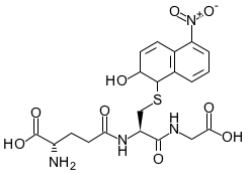
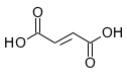
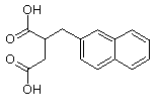
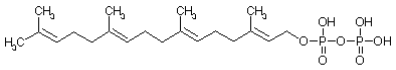
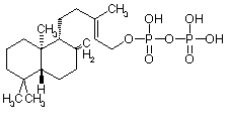
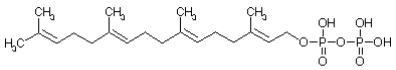
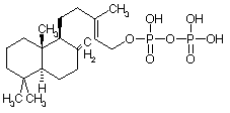
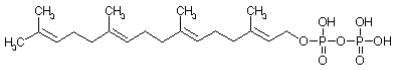
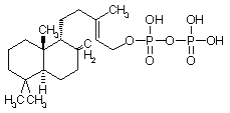
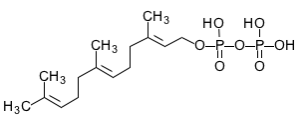
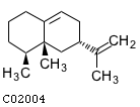
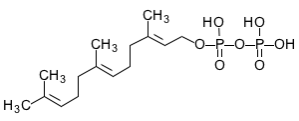
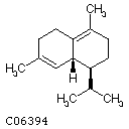
RP13013		
RP13014		

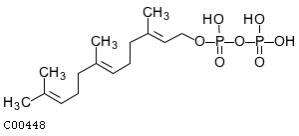
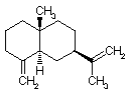
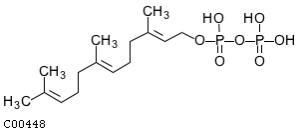
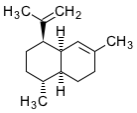
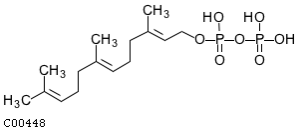
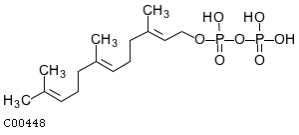
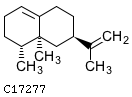
1-59

RP03116		
RP11492		

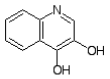
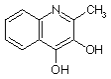
1-60

RP05255		
RP10729		
RP10077		
RP10078		
RP10079		

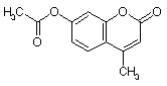
RP10080	 C00051	 C14803
RP10081	 C00051	 C14804
RP10082	 C00051	 C14805
RP10083	 C00051	 C14806
RP09161	 C00122	 C14115
RP01949	 C00353	 C06089
RP05493	 C00353	 C11901
RP11107	 C00353	 C11902
RP02118	 C00448	 C02004
RP02120	 C00448	 C06394

RP12983	 C00448	 C09723
RP11141	 C00448	 C16028
RP12984	 C00448	 C16286
RP13638	 C00448	 C17277

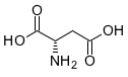
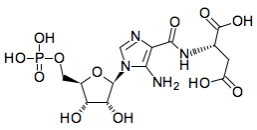
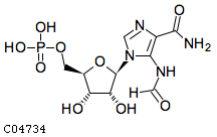
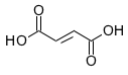
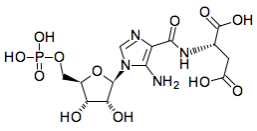
1-61

RP04520	 C00237	 C06330
RP04755	 C00237	 C06331

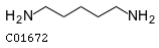
1-62

RP03770	 C00033	 C03837

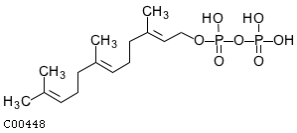
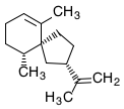
1-65

RP04240	 C00049	 C04823
RP10097	 C00058	 C04734
RP04203	 C00122	 C04823

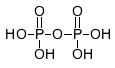
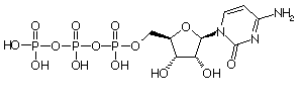
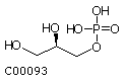
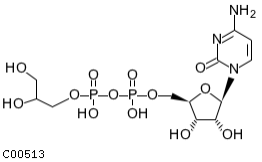
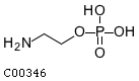
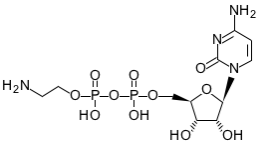
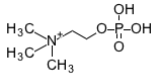
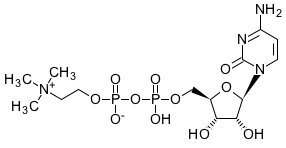
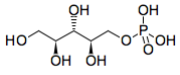
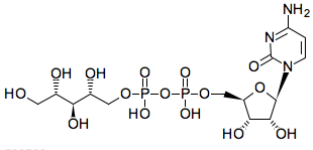
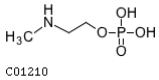
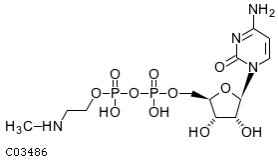
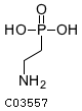
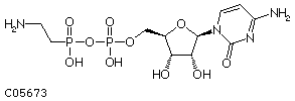
1-69

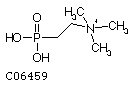
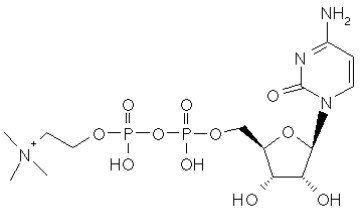
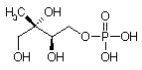
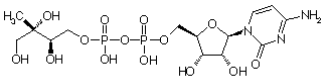
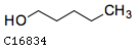
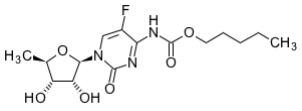
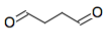
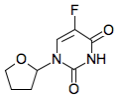
RP13020	 C01672	 C10773

1-70

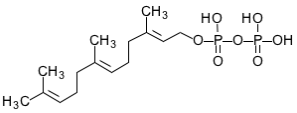
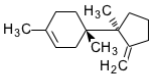
RP09032	 C00448	 C12142

1-73

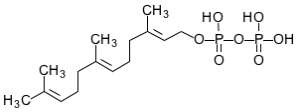
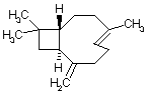
RP01194	 C00013	 C00063
RP00634	 C00093	 C00513
RP01928	 C00346	 C00570
RP01824	 C00588	 C00307
RP02604	 C01068	 C00789
RP02994	 C01210	 C03486
RP03880	 C03557	 C05673

RP02328	 C06459	 C05674
RP05238	 C11434	 C11435
RP13294	 C16834	 C12650
RP13299	 C16835	 C12673

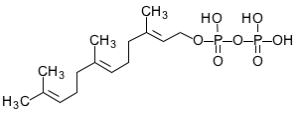
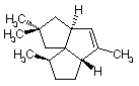
1-77

RP02117	 C00448	 C01860

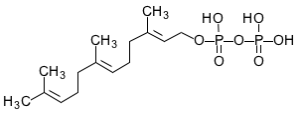
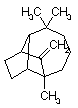
1-80

RP12978	 C00448	 C09629

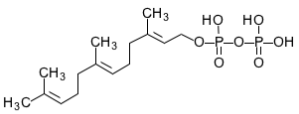
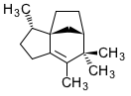
1-82

RP02116	 C00448	 C01841

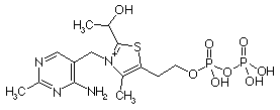
1-84

RP12981	 C00448	 C09699

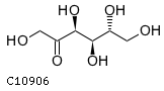
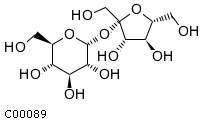
1-86

RP11146	 C00448	 C16269

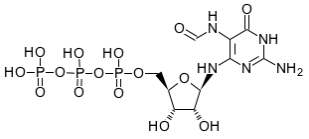
1-87

RP01089		

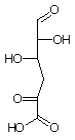
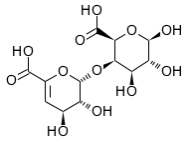
1-88

RP10922		

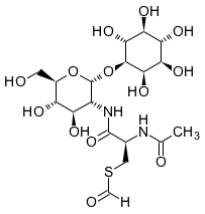
1-89

RP04649		

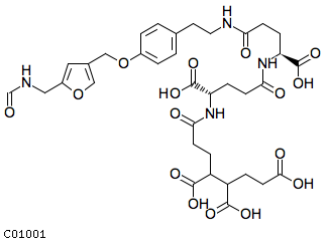
1-105

RP04026		

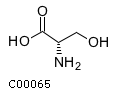
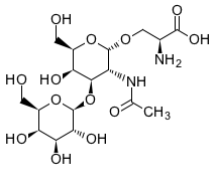
1-108

RP10859		

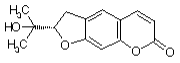
1-109

RP10840		

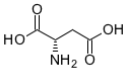
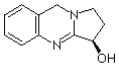
1-110

RP04170	 C00065	 C04776

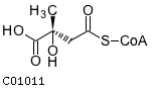
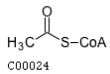
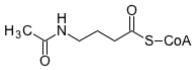
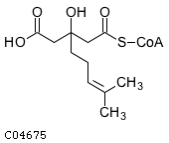
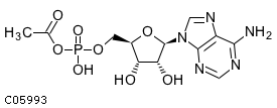
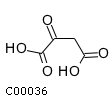
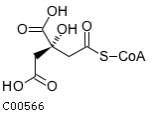
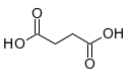
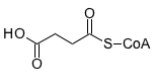
1-121

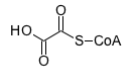
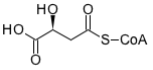
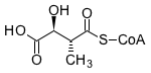
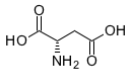
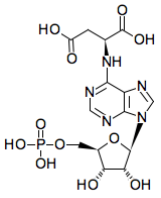
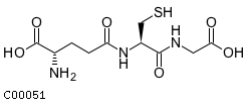
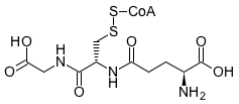
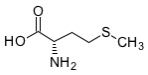
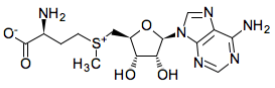
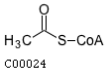
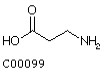
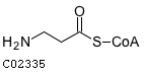
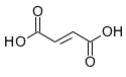
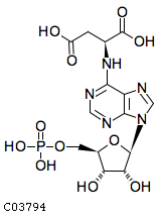
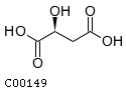
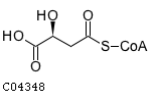
RP12948	 C00207	 C09276

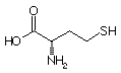
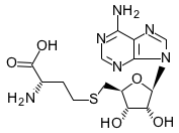
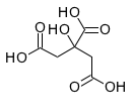
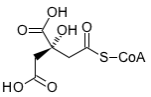
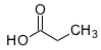
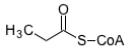
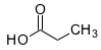
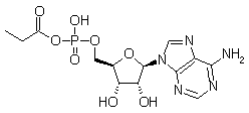
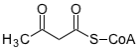
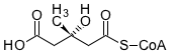
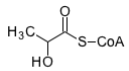
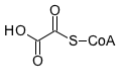
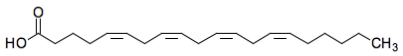
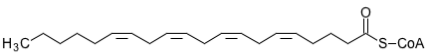
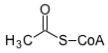
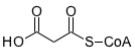
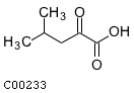
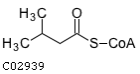
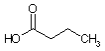
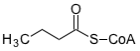
1-130

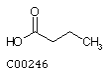
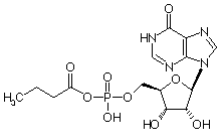
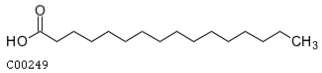
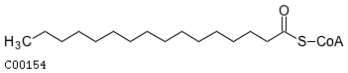
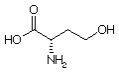
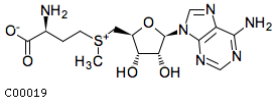
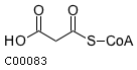
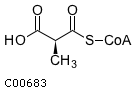
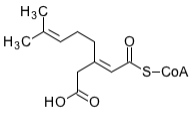
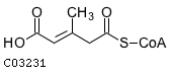
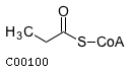
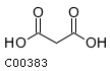
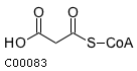
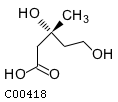
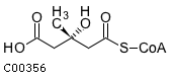
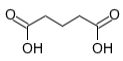
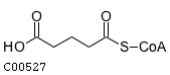
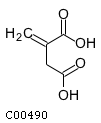
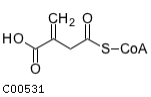
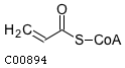
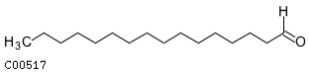
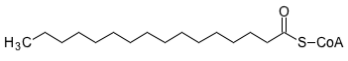
RP12917	 C00049	 C10733

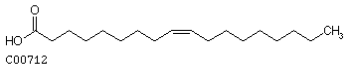
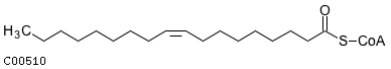
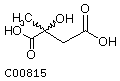
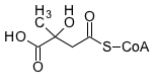
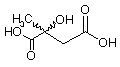
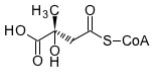
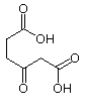
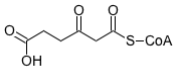
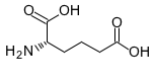
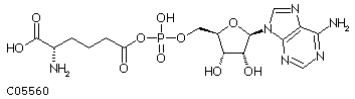
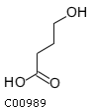
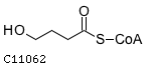
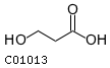
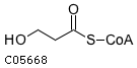
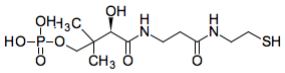
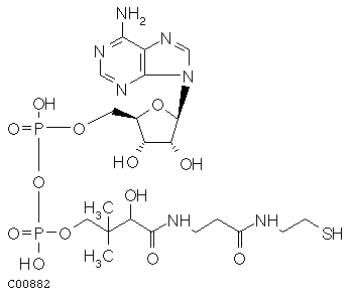
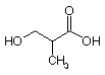
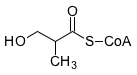
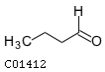
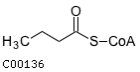
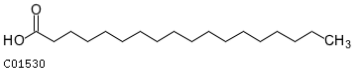
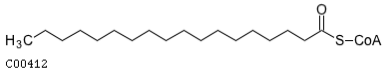
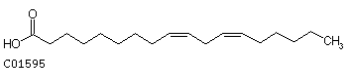
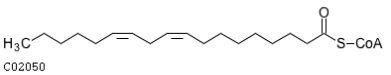
1-141

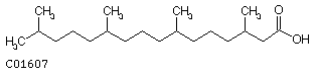
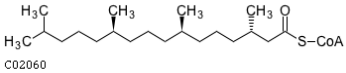
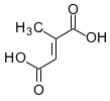
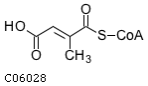
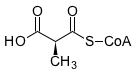
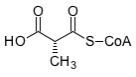
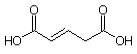
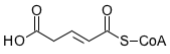
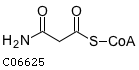
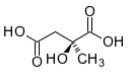
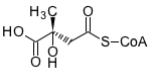
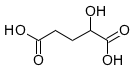
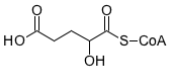
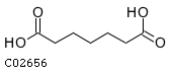
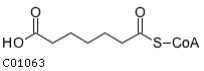
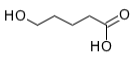
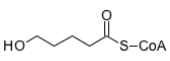
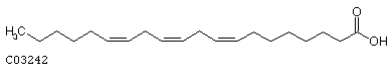
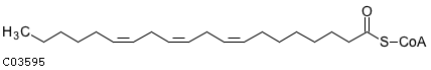
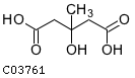
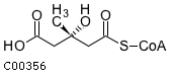
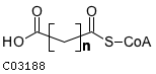
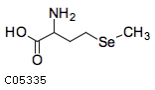
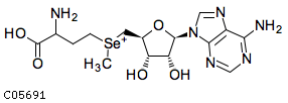
RP04364	 C00022	 C01011
RP00026	 C00033	 C00024
RP03681	 C00033	 C03357
RP03919	 C00033	 C04675
RP04833	 C00033	 C05993
RP05290	 C00036	 C00566
RP00032	 C00042	 C00091

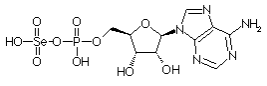
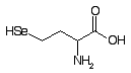
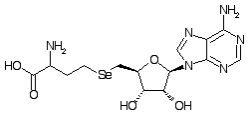
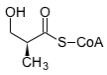
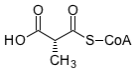
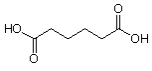
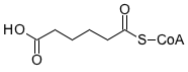
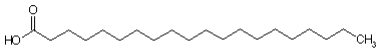
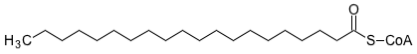
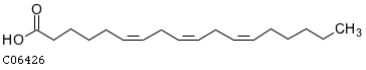
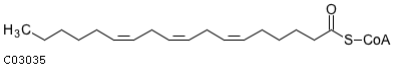
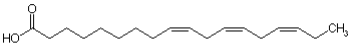
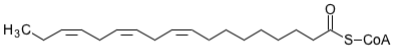
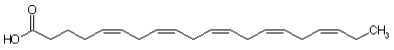
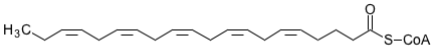
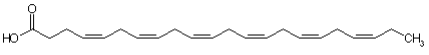
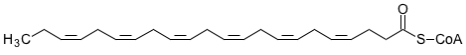
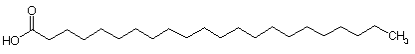
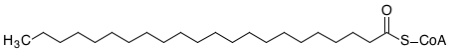
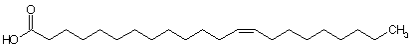
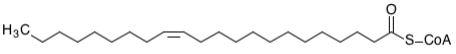
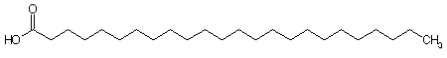
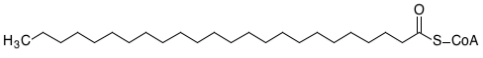
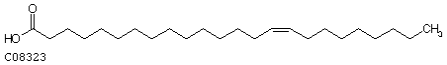
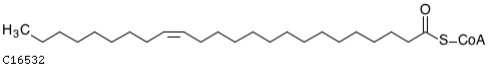
RP00916	 C00048	 C00313
RP00923	 C00048	 C04348
RP01183	 C00048	 C06027
RP01324	 C00049	 C03794
RP05310	 C00051	 C00920
RP00953	 C00058	 C00798
RP03934	 C00073	 C00019
RP04321	 C00084	 C00024
RP01176	 C00099	 C02335
RP01294	 C00122	 C03794
RP00378	 C00149	 C04348

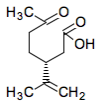
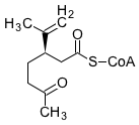
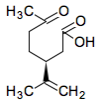
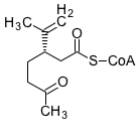
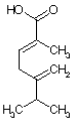
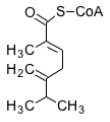
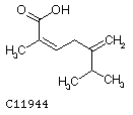
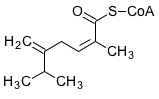
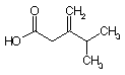
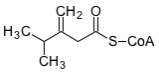
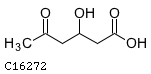
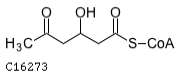
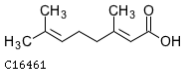
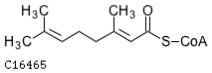
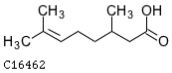
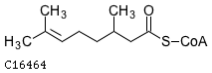
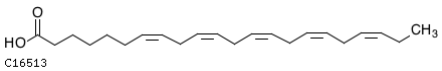
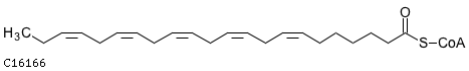
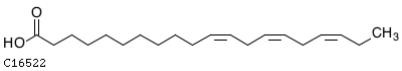
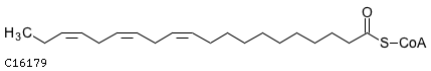
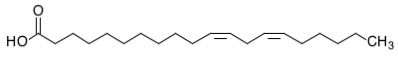
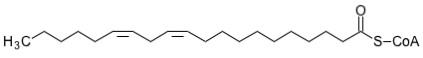
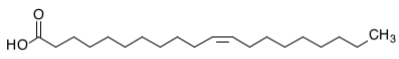
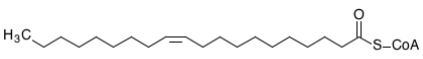
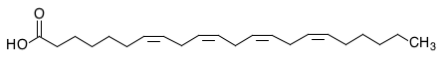
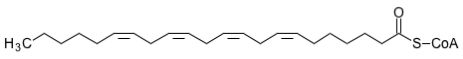
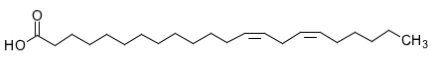
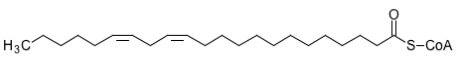
RP04107	 C00155	 C00021
RP00562	 C00158	 C00566
RP00117	 C00163	 C00100
RP01453	 C00163	 C05983
RP00085	 C00164	 C00332
RP01457	 C00164	 C00356
RP00368	 C00186	 C00827
RP00498	 C00209	 C00313
RP01625	 C00219	 C02249
RP00606	 C00222	 C00024
RP01076	 C00222	 C00083
RP01666	 C00233	 C02939
RP00206	 C00246	 C00136

RP01694		
RP00237		
RP03911		
RP01077		
RP01803		
RP03108		
RP03766		
RP01178		
RP01078		
RP01960		
RP02183		
RP00140		
RP05100		
RP01403		

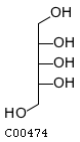
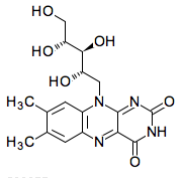
RP12987	 C00712	 C00510
RP02636	 C00815	 C00904
RP02635	 C00815	 C01011
RP02665	 C00846	 C02232
RP00497	 C00956	 C05560
RP04939	 C00989	 C11062
RP00479	 C01013	 C05668
RP00613	 C01134	 C00882
RP02969	 C01188	 C04047
RP00515	 C01412	 C00136
RP12974	 C01530	 C00412
RP13018	 C01595	 C02050

RP03244	 C01607	 C02060
RP03324	 C01732	 C06028
RP02481	 C02170	 C00683
RP03002	 C02170	 C01213
RP03501	 C02214	 C02411
RP05068	 C02362	 C06625
RP00401	 C02614	 C01011
RP03620	 C02630	 C03058
RP02845	 C02656	 C01063
RP03683	 C02804	 C03237
RP13039	 C03242	 C03595
RP01961	 C03761	 C00356
RP03790	 C04025	 C03188
RP04405	 C05335	 C05691

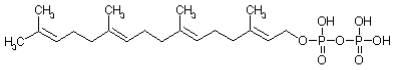
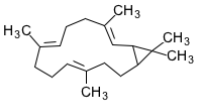
RP04533	 C05697	 C05686
RP04541	 C05698	 C05692
RP04667	 C06001	 C06000
RP03001	 C06002	 C01213
RP09373	 C06104	 C14143
RP13025	 C06425	 C02041
RP13038	 C06426	 C03035
RP13081	 C06427	 C16162
RP13082	 C06428	 C16165
RP13083	 C06429	 C16169
RP13161	 C08281	 C16528
RP13164	 C08316	 C16531
RP13165	 C08320	 C16529
RP13166	 C08323	 C16532

RP00860	 C11405	 C11407
RP05556	 C11419	 C11421
RP05572	 C11943	 C11945
RP05570	 C11944	 C11946
RP05578	 C11950	 C11949
RP12134	 C16272	 C16273
RP12224	 C16461	 C16465
RP12225	 C16462	 C16464
RP13199	 C16513	 C16166
RP13201	 C16522	 C16179
RP13202	 C16525	 C16180
RP13205	 C16526	 C16530
RP13200	 C16527	 C16170
RP13206	 C16533	 C16645

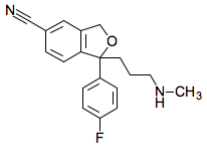
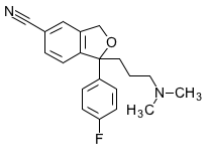
1-142

RP01722	 C00474	 C00255

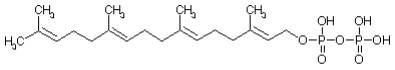
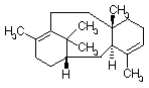
1-160

RP01947	 C00353	 C01414

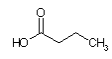
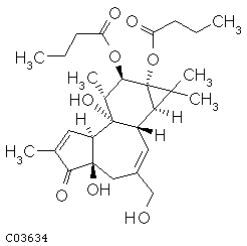
1-171

RP11053	 C00218	 C16608
RP10959	 C00543	 C07572

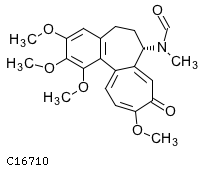
1-172

RP09176	 C00353	 C11894

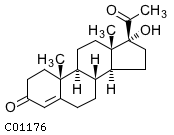
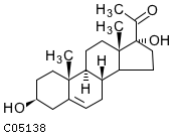
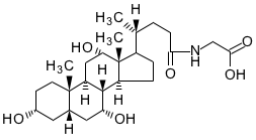
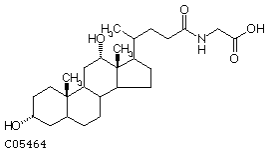
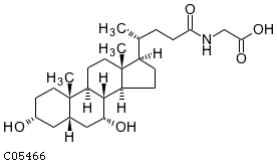
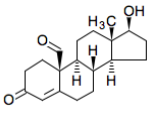
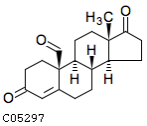
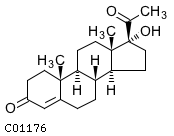
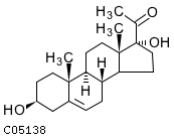
1-180

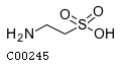
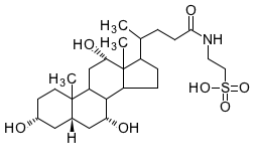
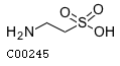
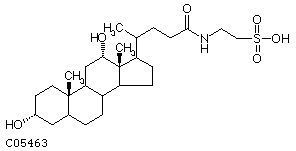
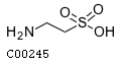
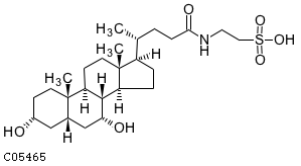
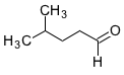
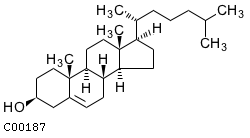
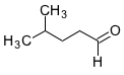
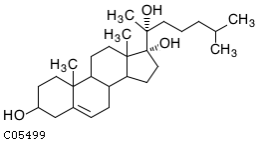
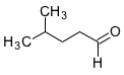
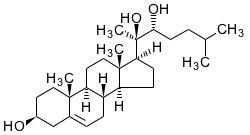
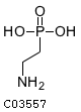
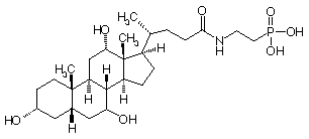
RP03748	 C00246	 C03634

1-188

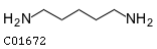
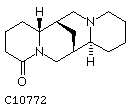
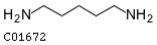
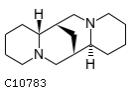
RP13027	 C00058	 C16710

1-211

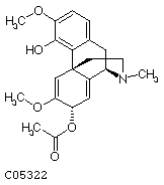
RP12714	 C00033	 C01176
RP03017	 C00033	 C05138
RP00707	 C00037	 C01921
RP04124	 C00037	 C05464
RP03593	 C00037	 C05466
RP12729	 C00058	 C05295
RP06976	 C00058	 C05297
RP01795	 C00084	 C01176
RP12715	 C00084	 C05138

RP00523	 C00245	 C05122
RP04126	 C00245	 C05463
RP03597	 C00245	 C05465
RP02438	 C02373	 C00187
RP08256	 C02373	 C05499
RP03553	 C02373	 C05501
RP11461	 C03557	 C05683

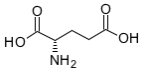
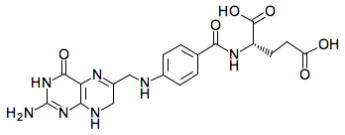
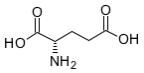
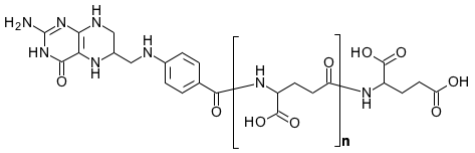
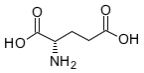
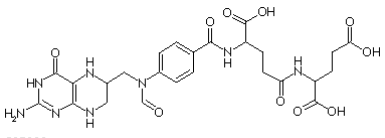
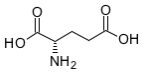
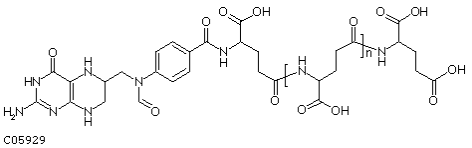
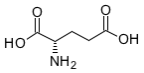
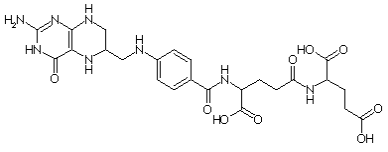
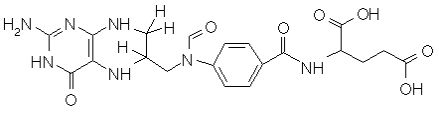
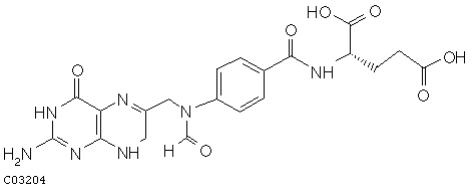
1-215

RP13019	 C01672	 C10772
RP13021	 C01672	 C10783

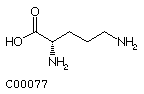
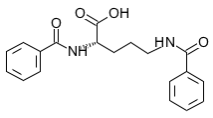
1-216

RP12446	 C00033	 C05322

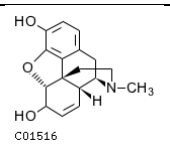
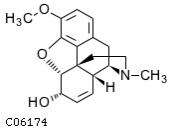
1-223

RP02060	 C00025	 C00415
RP03875	 C00025	 C03541
RP01668	 C00025	 C05928
RP04795	 C00025	 C05929
RP01186	 C00025	 C09332
RP00300	 C00058	 C00234
RP02062	 C00058	 C03204

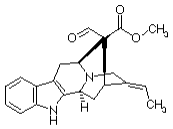
1-253

RP01038	 C00077	 C03712

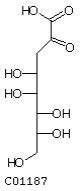
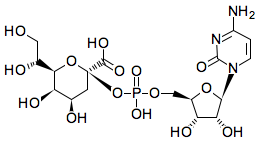
1-254

RP13310	 C00067	 C01516
RP12618	 C00067	 C06174

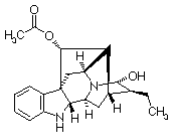
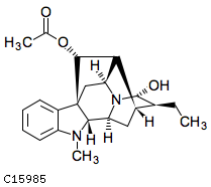
1-259

RP09163	C00132	 C11632

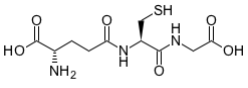
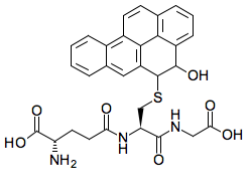
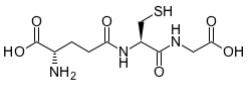
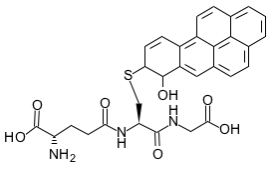
1-282

RP02966	 C01187	 C04121

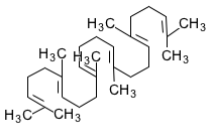
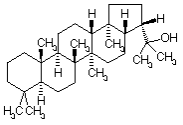
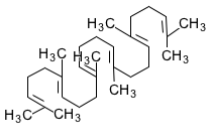
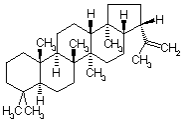
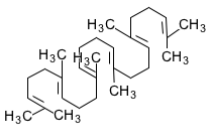
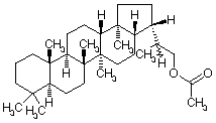
1-288

RP09145	 C00033	 C11809
RP10815	 C00033	 C15985

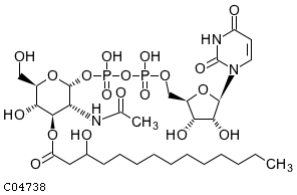
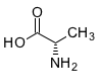
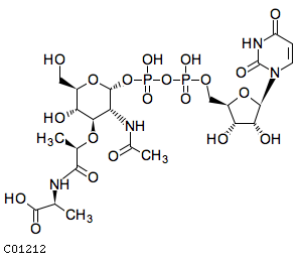
1-290

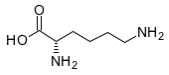
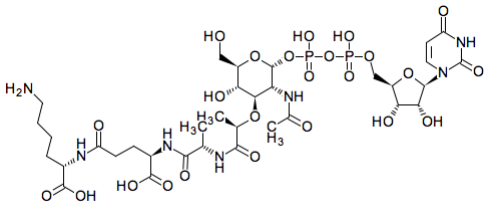
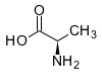
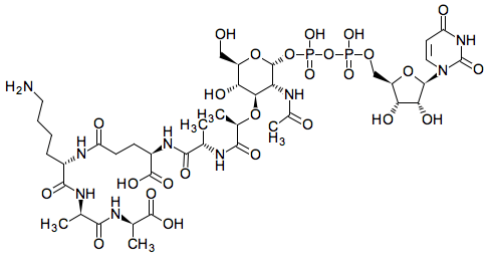
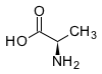
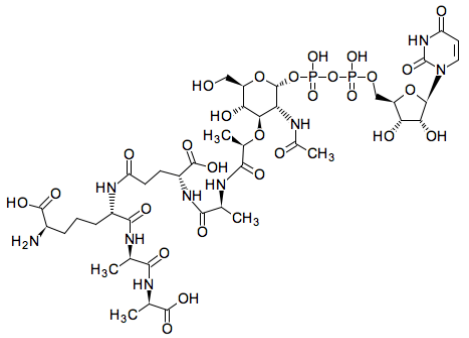
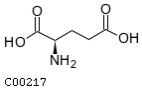
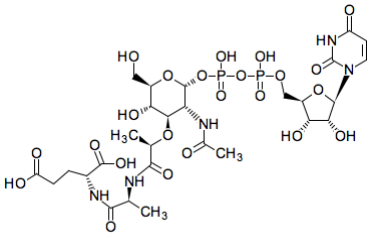
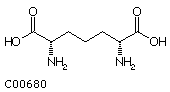
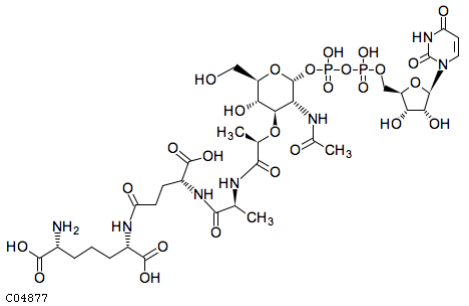
RP10086	 C00051	 C14855
RP10087	 C00051	 C14856

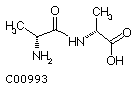
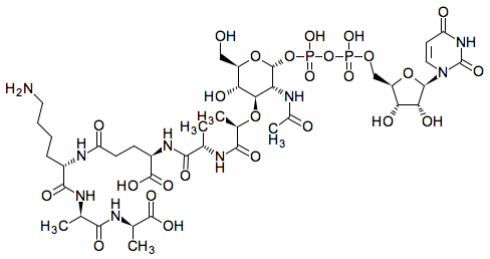
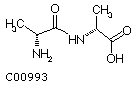
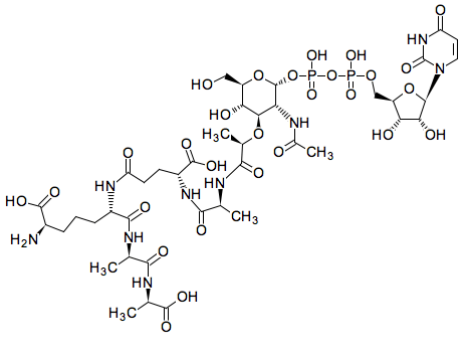
1-311

RP11222	 C00751	 C06309
RP11223	 C00751	 C06310
RP12957	 C00751	 C08627

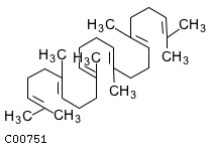
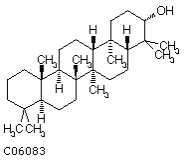
1-317

RP04234	 C00033	 C04738
RP02828	 C00041	 C01212

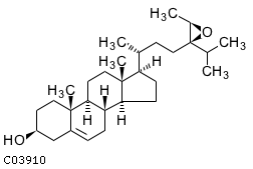
RP02498	 C00047	 C05892
RP04262	 C00133	 C04702
RP13910	 C00133	 C04882
RP02494	 C00217	 C00692
RP02500	 C00680	 C04877

RP04219		
RP04266		

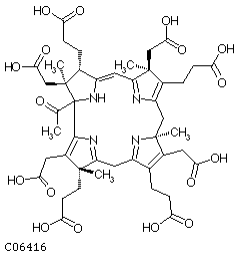
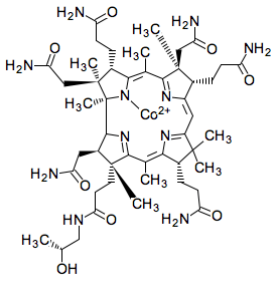
1-324

RP09205		

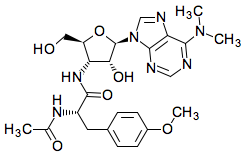
1-333

RP09156		

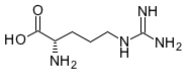
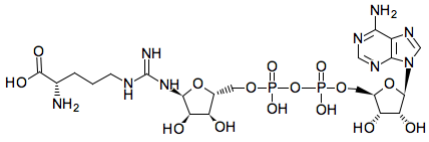
1-334

RP08529		
RP04584		

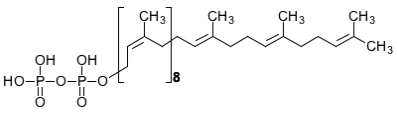
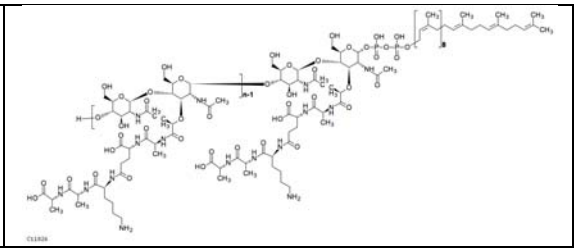
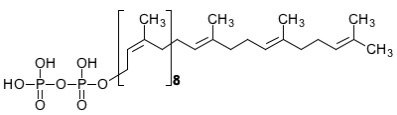
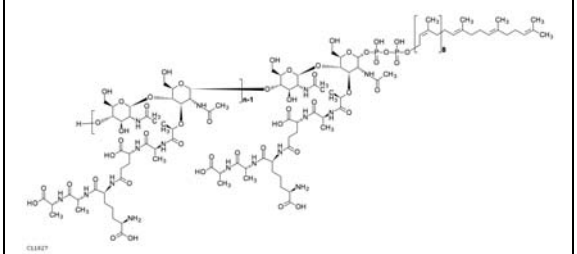
1-336

RP05284		

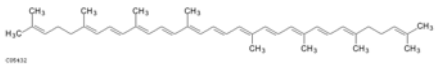
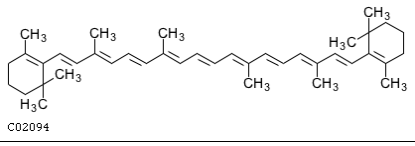
1-371

RP00492		

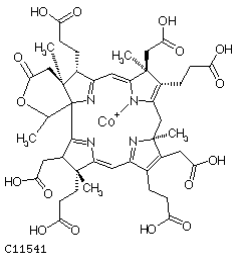
1-382

RP13905		
RP13902		

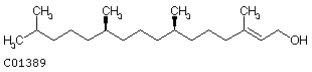
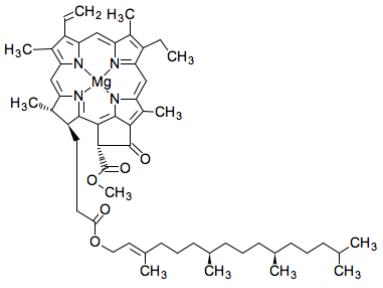
1-397

RP11366		

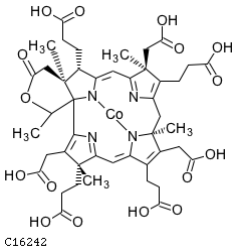
1-403

RP08882		

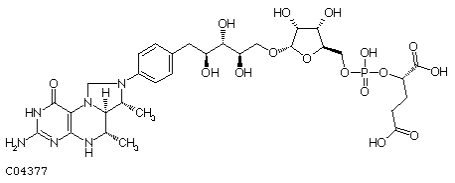
1-408

RP05219		

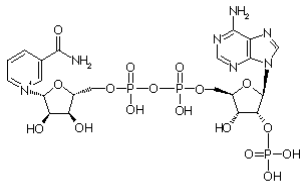
1-427

RP10913		

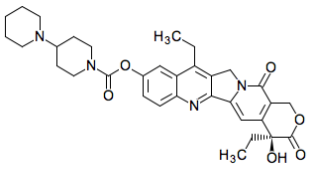
1-430

RP10858		

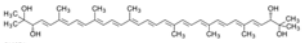
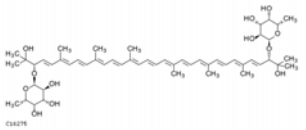
1-444

RP03619		

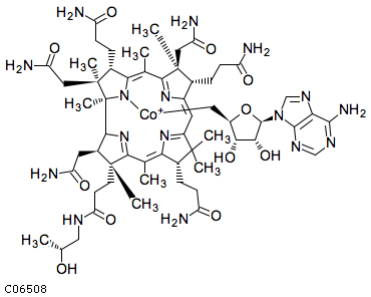
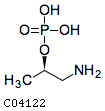
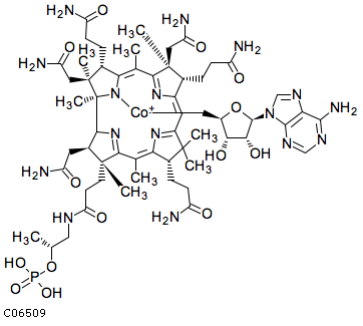
1-468

RP13307		

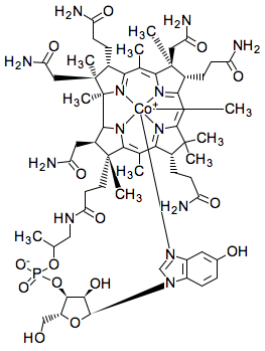
1-488

RP12135		

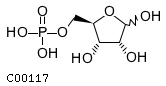
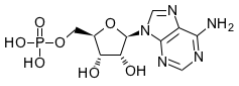
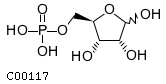
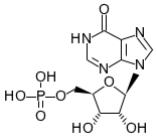
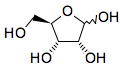
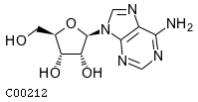
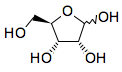
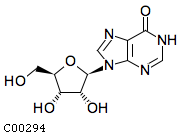
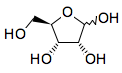
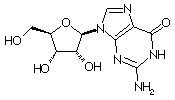
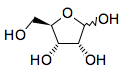
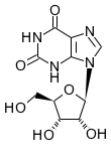
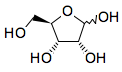
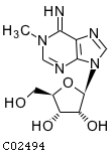
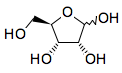
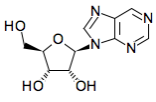
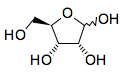
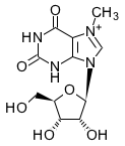
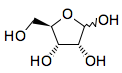
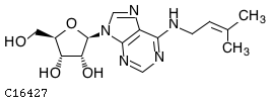
1-500

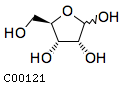
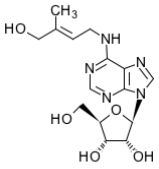
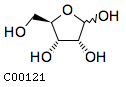
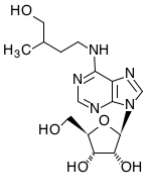
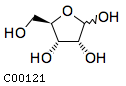
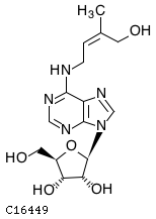
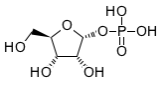
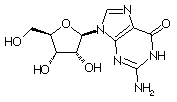
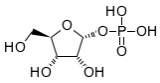
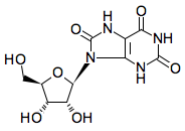
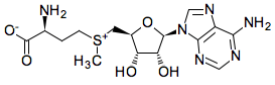
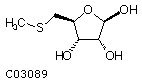
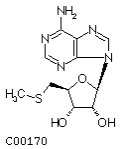
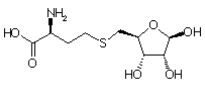
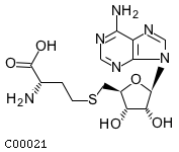
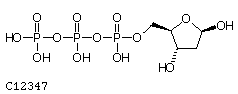
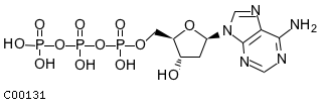
RP11445		
RP05674		

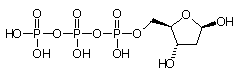
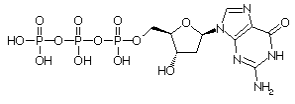
1-520

RP04029	H_3C-OH C00132	 C04665

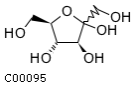
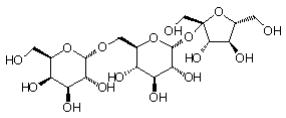
6-141

RP04030	 C00117	 C00020
RP01320	 C00117	 C00130
RP01385	 C00121	 C00212
RP01744	 C00121	 C00294
RP01683	 C00121	 C00387
RP01996	 C00121	 C01762
RP03502	 C00121	 C02494
RP10963	 C00121	 C15586
RP10964	 C00121	 C16352
RP10965	 C00121	 C16427

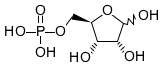
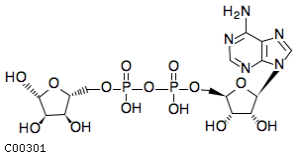
RP10966	 C00121	 C16431
RP10967	 C00121	 C16447
RP10968	 C00121	 C16449
RP02001	 C00620	 C00387
RP02376	 C00620	 C05513
RP03959	 C02926	 C00019
RP01483	 C03089	 C00170
RP04184	 C03539	 C00021
RP09162	 C12347	 C00131

RP11060		

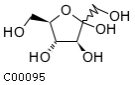
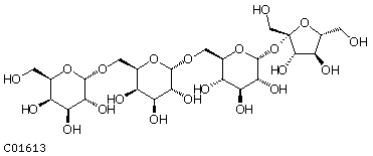
6-266

RP02187		

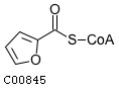
6-371

RP00384		

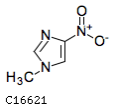
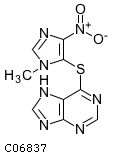
6-392

RP03248		

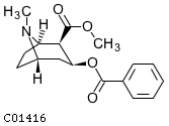
6-470

RP02663		

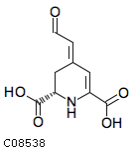
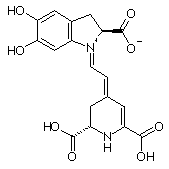
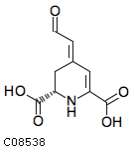
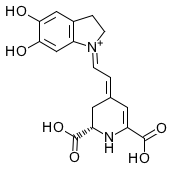
7-169

RP13099		

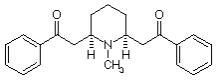
8-185

RP09233		

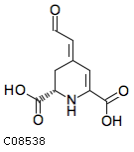
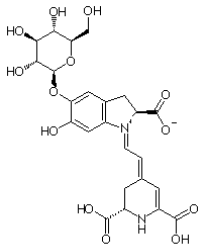
14-204

RP13811	 C08538	 C08539
RP13832	 C08538	 C17757

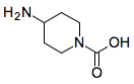
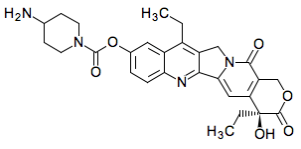
14-300

RP13074	 C06181	 C10157

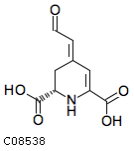
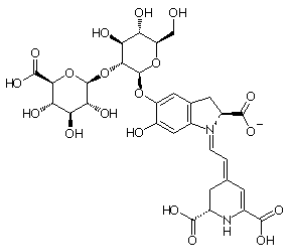
14-339

RP13810	 C08538	 C08540

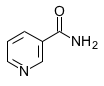
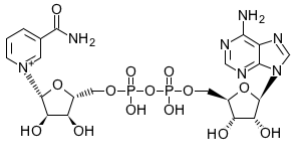
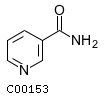
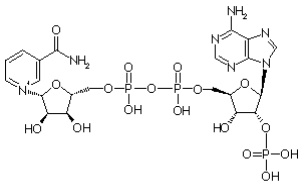
14-428

RP13305	 C16837	 C16543

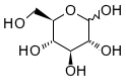
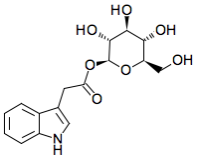
14-437

RP13807	 C08538	 C08537

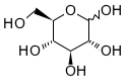
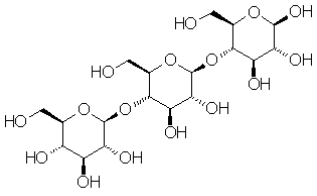
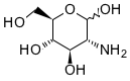
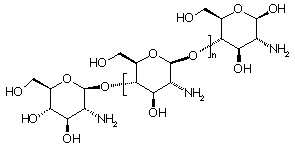
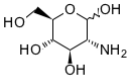
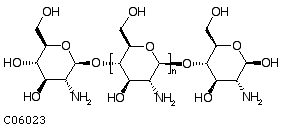
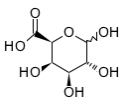
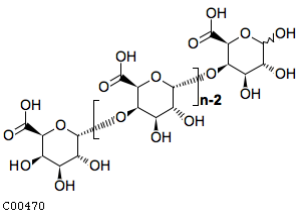
14-444

RP00082	 C00153	 C00003
RP03626	 C00153	 C00006

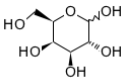
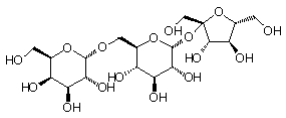
15-226

RP03967	 C00031	 C04197

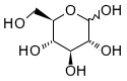
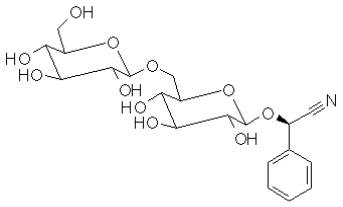
15-265

RP01512	 C00031	 C06219
RP13717	 C00329	 C00734
RP01879	 C00329	 C06023
RP01890	 C00333	 C00470

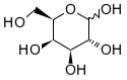
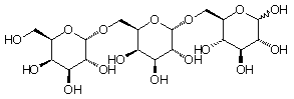
15-266

RP01308	 C00124	 C00492

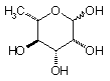
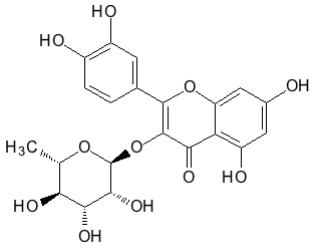
15-298

RP02660	 C00031	 C01982

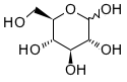
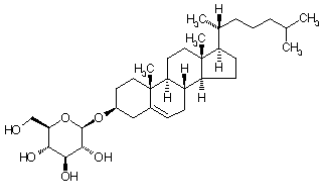
15-299

RP05135	 C00124	 C05404

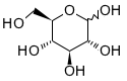
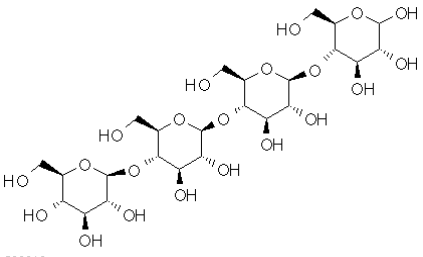
15-326

RP02214	 C00507	 C01750

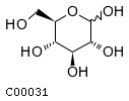
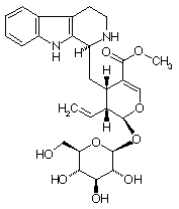
15-345

RP01524	 C00031	 C03855

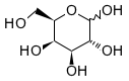
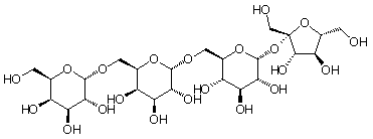
15-382

RP03423	 C00031	 C02013

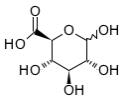
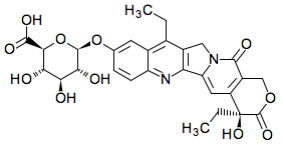
15-391

RP09143	 C00031	 C03470

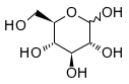
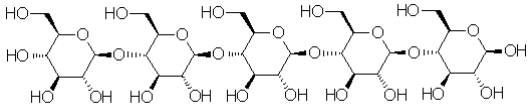
15-392

RP03247	 C00124	 C01613

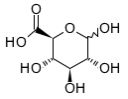
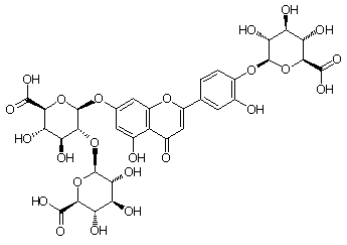
15-412

RP12946	 C00191	 C11376

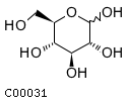
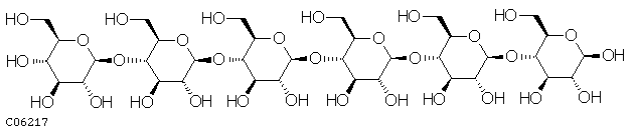
15-457

RP03425	 C00031	 C06218

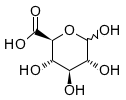
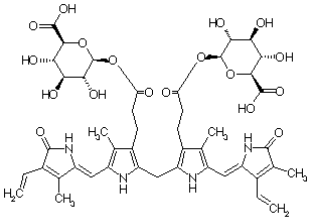
15-476

RP11006	 C00191	 C04900

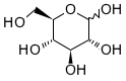
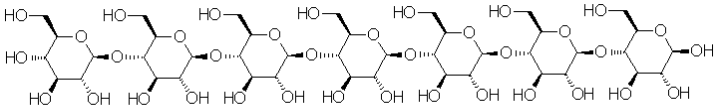
15-501

RP04740	 C00031	 C06217

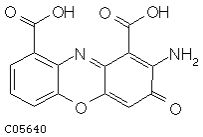
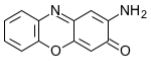
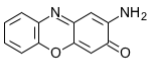
15-510

RP04588	 C00191	 C05787

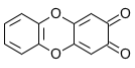
15-521

RP04738	 C00031	 C06216

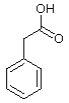
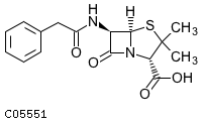
16-145

RP02397	 C00632	 C05640
RP03341	 C01987	 C02161
RP13508	 C17500	 C02161

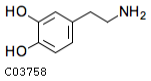
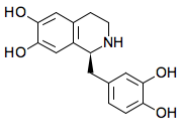
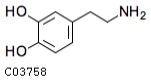
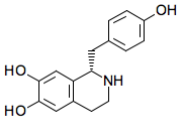
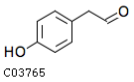
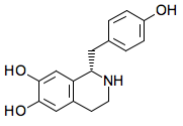
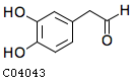
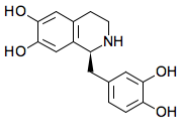
16-149

RP03385	 C00090	 C03861

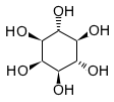
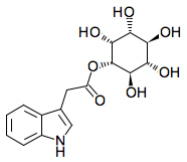
16-187

RP03729	 C07086	 C05551

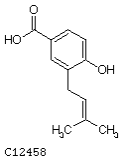
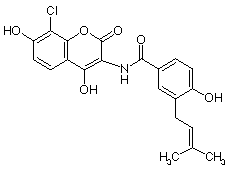
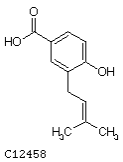
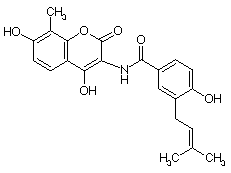
16-201

RP03710	 C03758	 C02916
RP03938	 C03758	 C06160
RP03939	 C03765	 C06160
RP03711	 C04043	 C02916

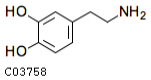
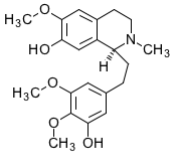
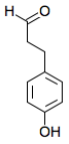
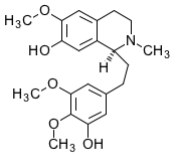
16-225

RP03966	 C00137	 C03868

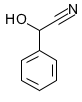
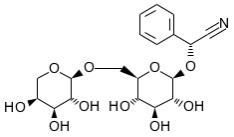
16-231

RP09549	 C12458	 C12471
RP09550	 C12458	 C12474

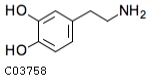
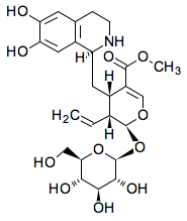
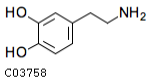
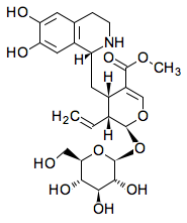
16-232

RP13047	 C03758	 C16707
RP13266	 C16706	 C16707

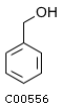
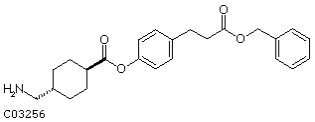
16-298

RP03255	 C00561	 C01870

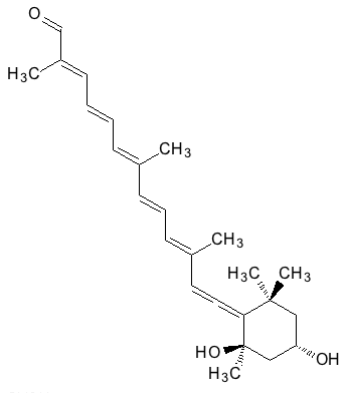
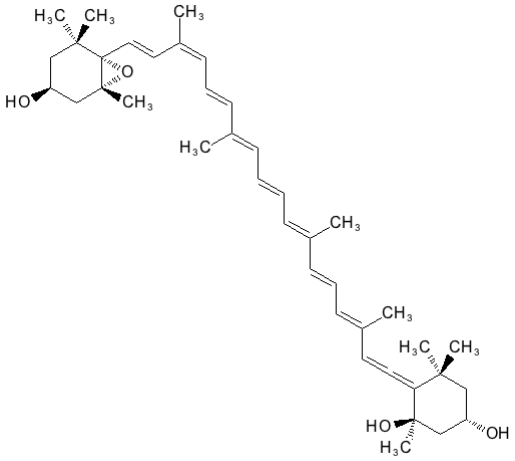
16-338

RP05338	 C03758	 C07304
RP05336	 C03758	 C07307

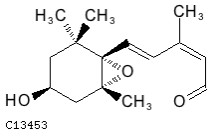
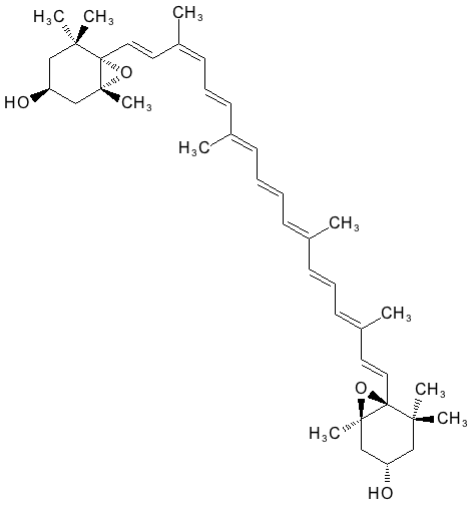
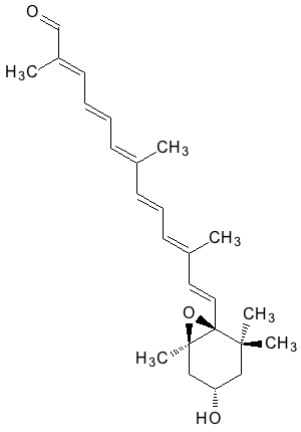
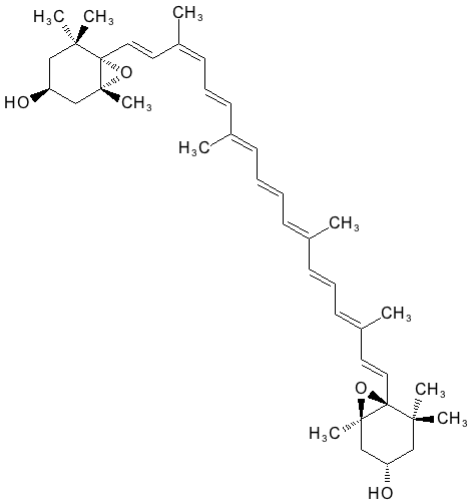
16-341

RP03222	 C00556	 C03256

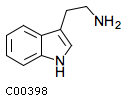
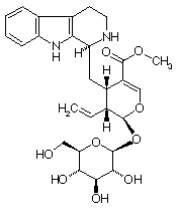
16-424

RP09595	 C14044	 C13431

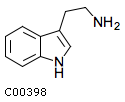
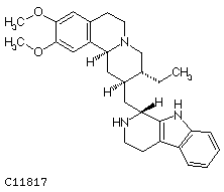
33-438

RP09596	 C13453	 C13433
RP09597	 C14045	 C13433

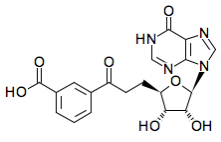
46-391

RP09180		

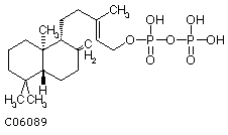
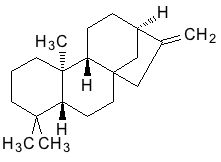
46-411

RP11124		

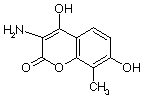
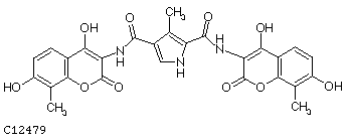
47-331

RP13546		

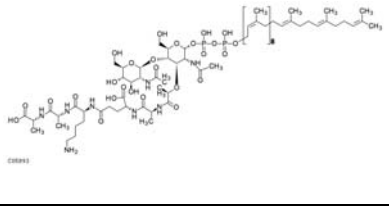
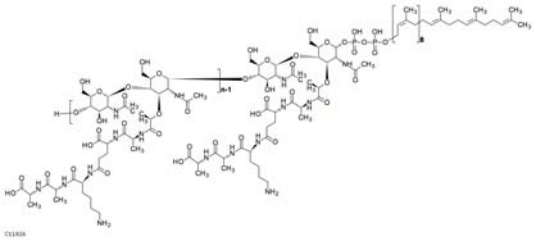
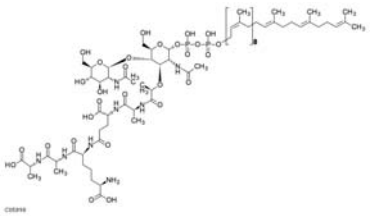
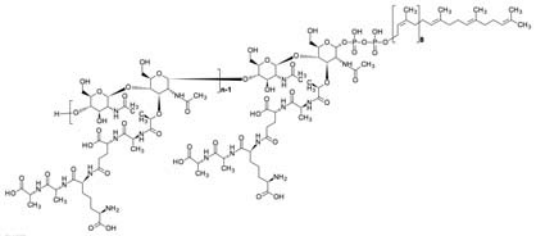
60-195

RP04693		

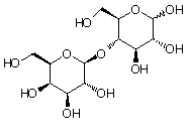
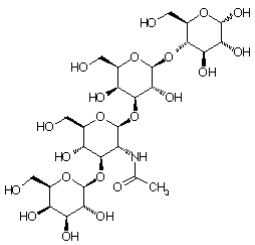
62-419

RP09561		

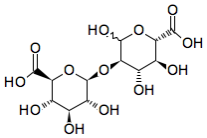
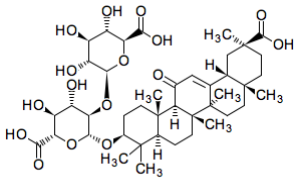
105-382

RP13904		
RP13901		

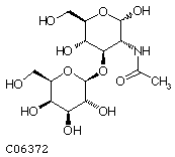
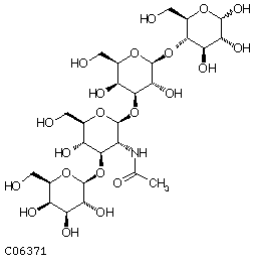
105-387

RP04768		

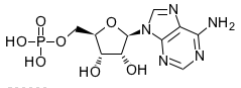
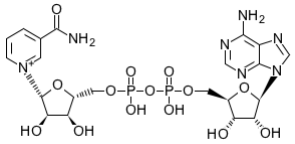
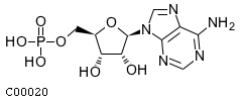
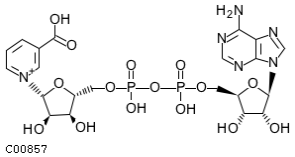
105-472

RP03525		

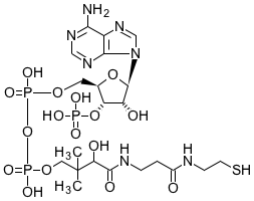
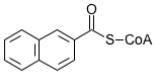
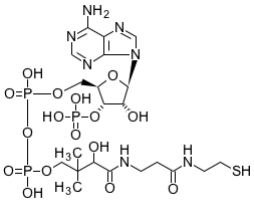
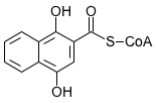
110-387

RP04767		

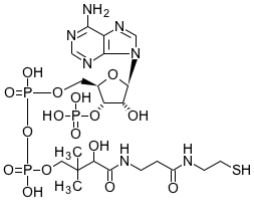
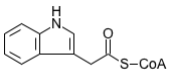
141-444

RP03498	 C00020	 C00003
RP02679	 C00020	 C00857

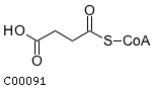
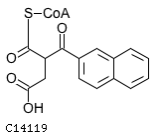
141-507

RP09290	 C00010	 C14120
RP10629	 C00010	 C15547

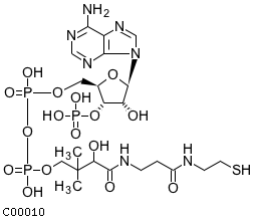
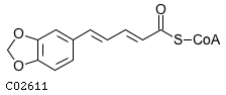
141-508

RP10627	 C00010	 C15489

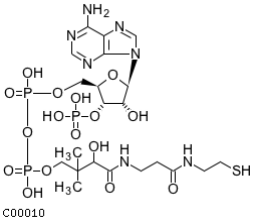
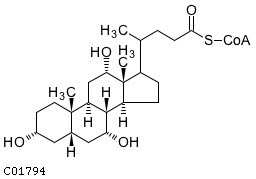
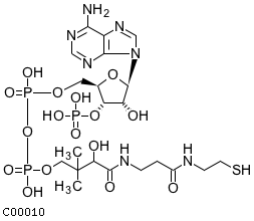
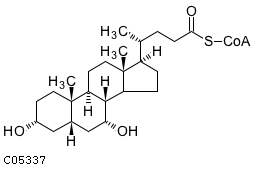
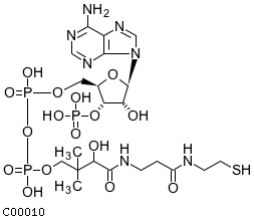
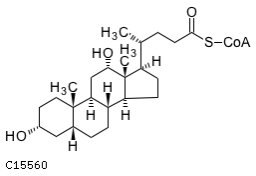
141-515

RP09157	 C00091	 C14119

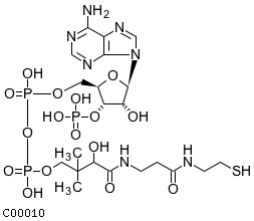
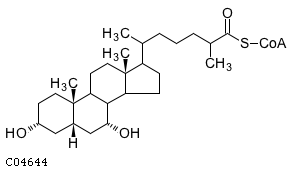
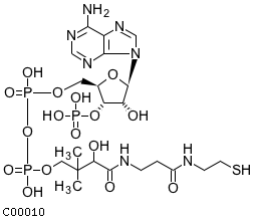
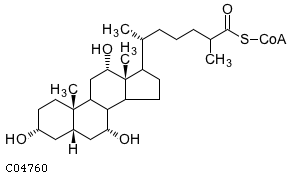
141-517

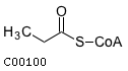
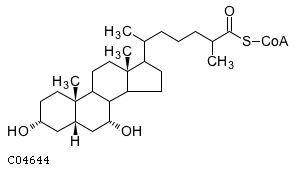
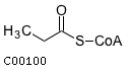
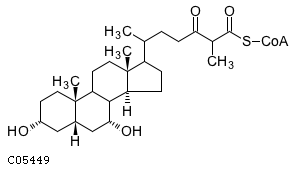
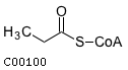
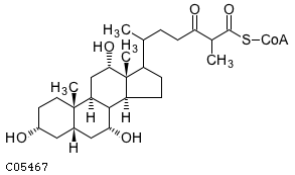
RP03615		

141-527

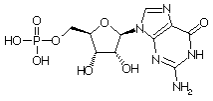
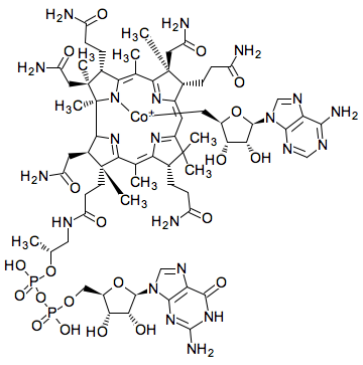
RP00192		
RP03591		
RP10630		

141-529

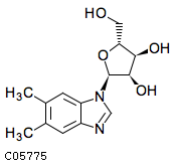
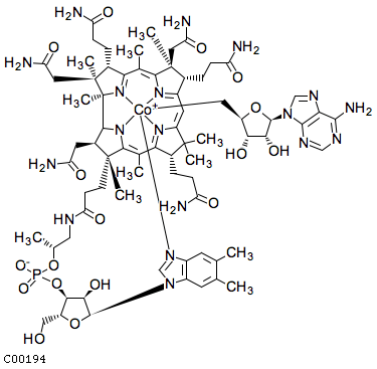
RP09271		
RP04225		

RP09815		
RP08317		
RP07683		

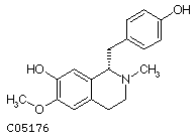
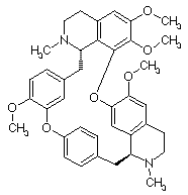
141-535

RP04822		

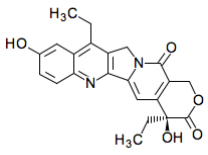
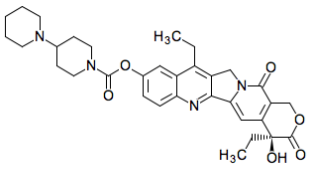
151-534

RP04823		

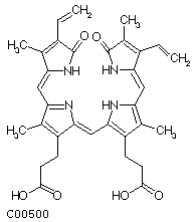
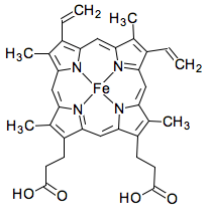
201-474

RP13061	 C05176	 C09654

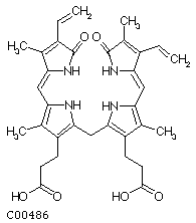
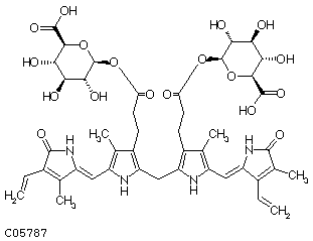
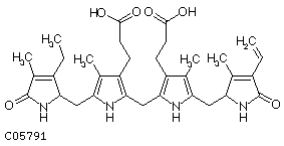
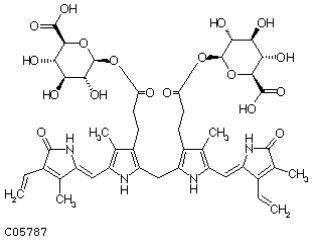
309-468

RP13188	 C11173	 C16641

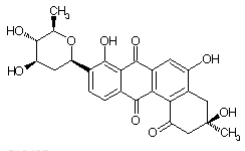
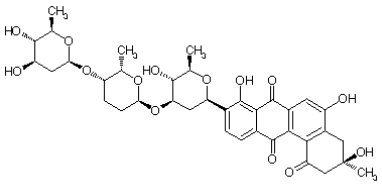
325-374

RP04716	 C00500	 C00032

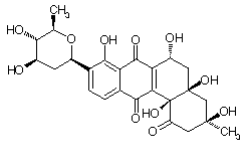
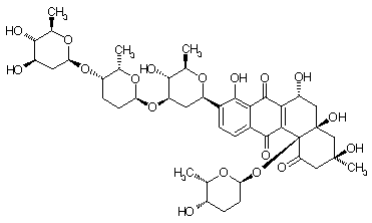
325-510

RP02176	 C00486	 C05787
RP04589	 C05791	 C05787

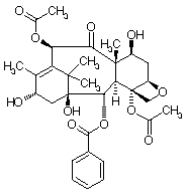
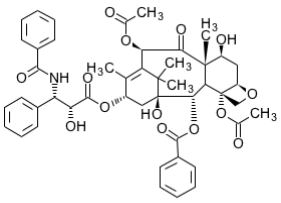
349-490

RP09518	 C12407	 C12409

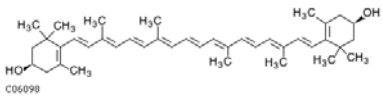
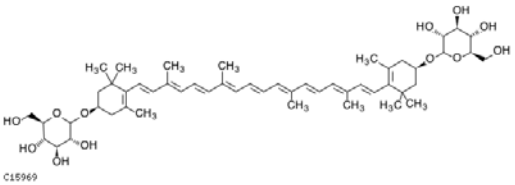
349-518

RP09521	 C12410	 C12411

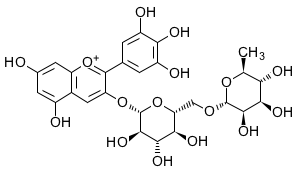
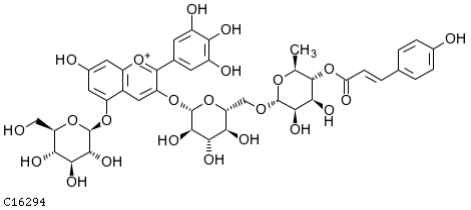
367-512

RP11715	 C11900	 C07394

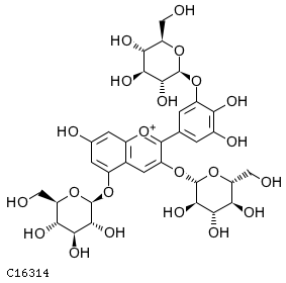
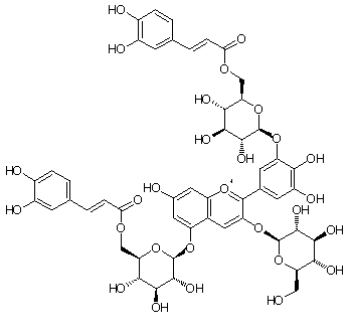
397-509

RP11646	 C06098	 C15969

435-523

RP12146	 C16315	 C16294

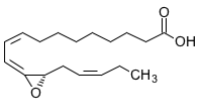
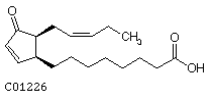
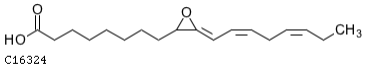
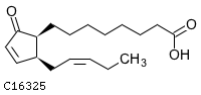
481-531

RP11755	 C16314	 C08641

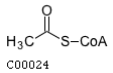
付録 C

KEGG RPAIR main に含まれる代謝物ペアの内、提案手法により Framework レベルの代謝経路予測を行った結果、Framework 間に包含関係の得られなかった代謝物ペア。各表上部の番号は Framework ID を意味している。表一列目の ID は KEGG RPAIR ID である。

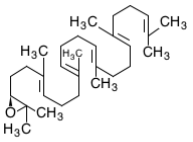
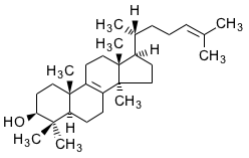
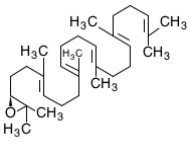
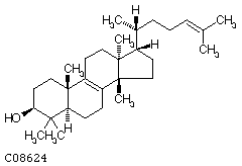
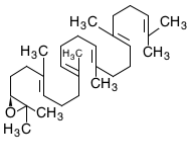
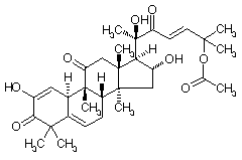
2-11

RP03016		
RP12161		

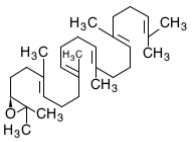
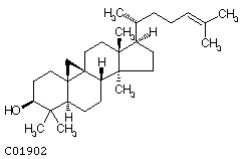
2-141

RP04948		

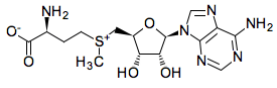
2-211

RP02834		
RP13004		
RP13005		

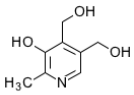
2-255

RP09218		

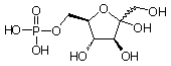
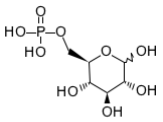
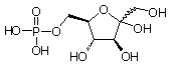
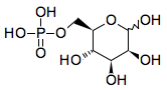
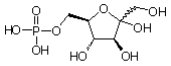
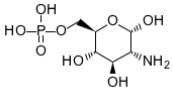
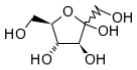
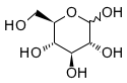
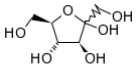
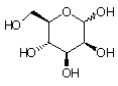
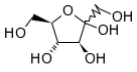
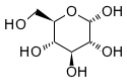
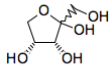
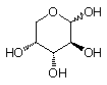
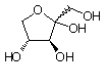
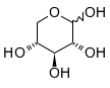
3-141

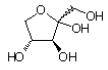
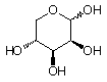
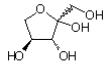
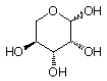
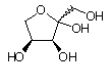
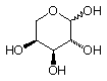
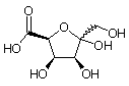
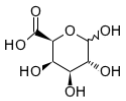
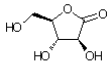
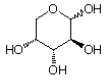
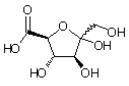
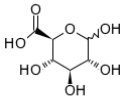
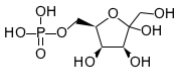
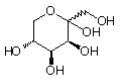
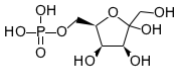
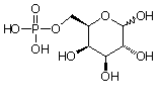
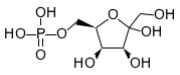
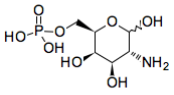
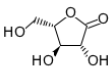
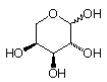
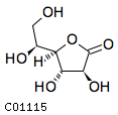
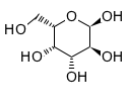
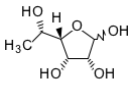
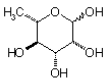
RP03942	 C01234	 C00019

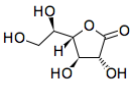
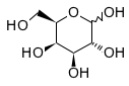
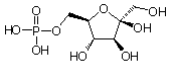
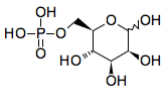
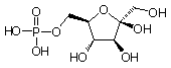
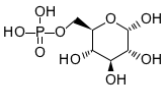
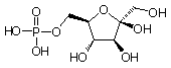
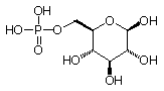
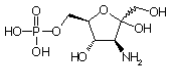
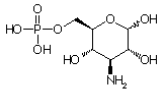
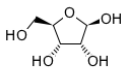
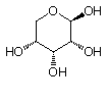
6-14

RP01840	 C06257	 C00314

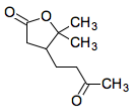
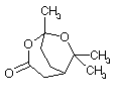
6-15

RP01093	 C00085	 C00092
RP01094	 C00085	 C00275
RP00434	 C00085	 C00352
RP04656	 C00095	 C00031
RP01153	 C00095	 C00159
RP01154	 C00095	 C00267
RP01604	 C00309	 C00216
RP01508	 C00310	 C00181

RP01828	 C00310	 C00476
RP01835	 C00312	 C01508
RP01736	 C00508	 C00259
RP01891	 C00558	 C00333
RP00525	 C00652	 C00216
RP01545	 C00905	 C00191
RP02609	 C01097	 C00795
RP02869	 C01097	 C01113
RP13006	 C01097	 C06377
RP01734	 C01114	 C00259
RP11272	 C01115	 C01825
RP02216	 C02431	 C00507

RP00430	 C03383	 C00124
RP01780	 C05345	 C00275
RP02454	 C05345	 C00668
RP02940	 C05345	 C01172
RP09107	 C12214	 C12213
RP13167	 C16639	 C08353

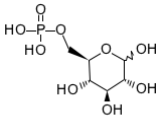
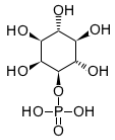
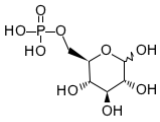
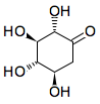
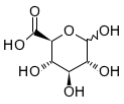
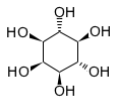
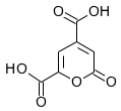
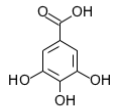
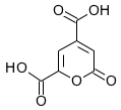
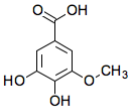
6-53

RP12483	 C06066	 C04718

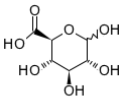
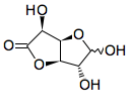
11-15

RP02303	 C00557	 C02240

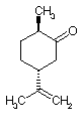
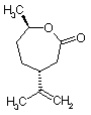
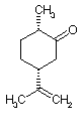
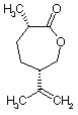
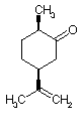
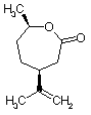
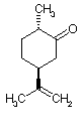
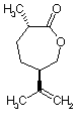
15-16

RP10933	 C00092	 C04006
RP13558	 C00092	 C17209
RP01348	 C00191	 C00137
RP03163	 C03671	 C01424
RP03912	 C03671	 C05616

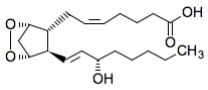
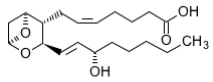
15-41

RP01546	 C00191	 C02670

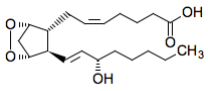
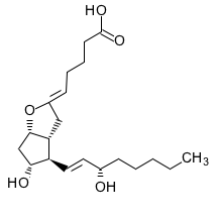
16-24

RP02055	 C00414	 C01880
RP02911	 C01147	 C03241
RP09043	 C01147	 C12314
RP05545	 C11398	 C11402
RP05547	 C11401	 C11403
RP05546	 C11412	 C11414
RP05548	 C11415	 C11418

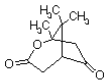
27-29

RP02087	 C00427	 C02198

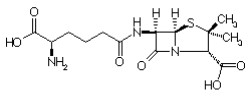
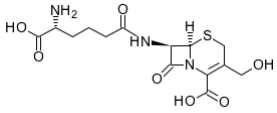
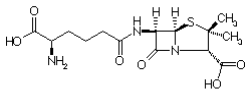
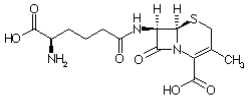
27-40

RP02086	 C00427	 C01312

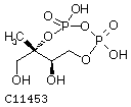
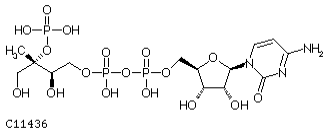
28-42

RP03727	 C03037	 C02952

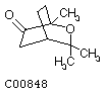
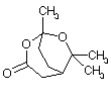
30-39

RP04829	 C06564	 C03112
RP04903	 C06564	 C06565

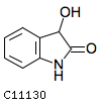
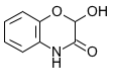
35-73

RP05245	 C11453	 C11436

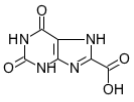
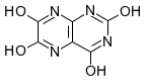
38-53

RP02669	 C00848	 C04718

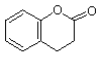
46-67

RP11832	 C11130	 C15769

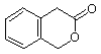
47-59

RP03787	 C03314	 C03178

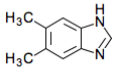
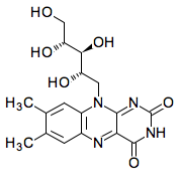
49-62

RP00834	 C01504	 C02274

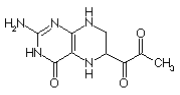
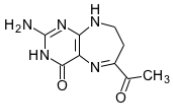
49-64

RP00839	 C07727	 C07728

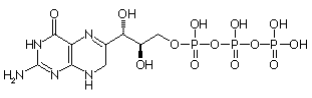
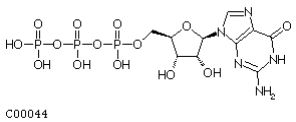
50-142

RP01724	 C03114	 C00255

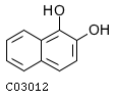
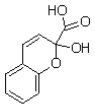
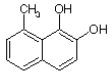
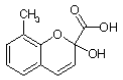
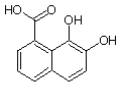
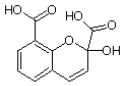
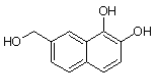
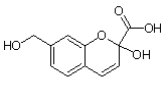
59-78

RP03605	 C03684	 C02587

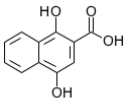
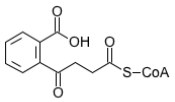
59-141

RP00891	 C04895	 C00044

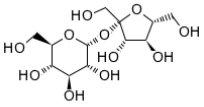
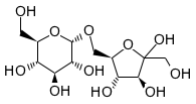
60-62

RP03745	 C03012	 C06204
RP09618	 C14084	 C14085
RP09626	 C14093	 C14094
RP09637	 C14105	 C14106

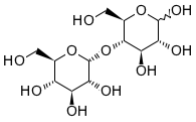
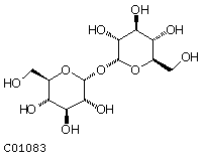
60-498

RP03783	 C03657	 C03160

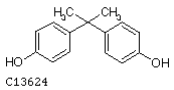
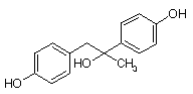
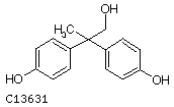
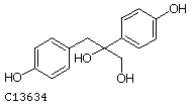
88-111

RP01113	 C00089	 C01742

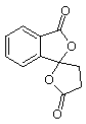
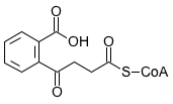
105-107

RP11015	 C00208	 C01083

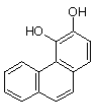
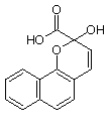
106-134

RP09607	 C13624	 C13629
RP09612	 C13631	 C13634

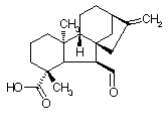
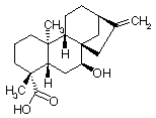
126-498

RP11441	 C06986	 C03160

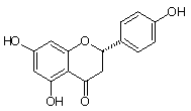
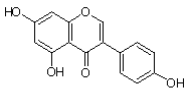
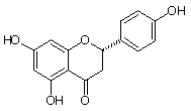
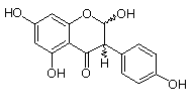
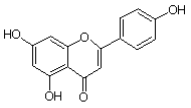
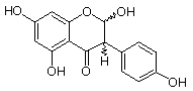
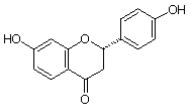
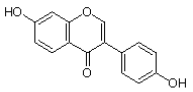
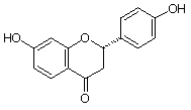
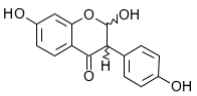
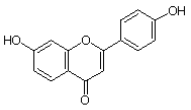
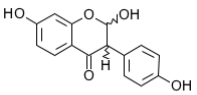
144-154

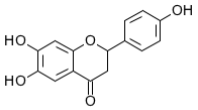
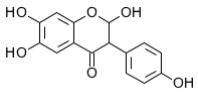
RP05258	 C03164	 C11425

179-195

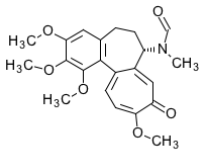
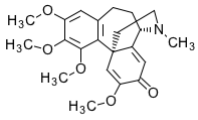
RP09369	 C06093	 C11875

183-184

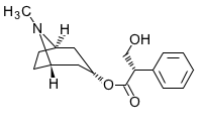
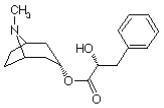
RP09033	 C00509	 C06563
RP11162	 C00509	 C12631
RP09239	 C01477	 C12631
RP09083	 C09762	 C10208
RP11791	 C09762	 C15567
RP11890	 C12123	 C15567

RP12118	 C16232	 C16233

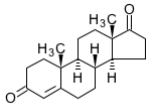
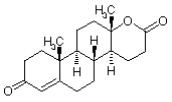
188-249

RP13270	 C16710	 C16709

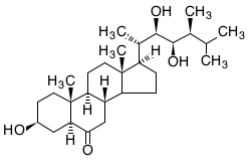
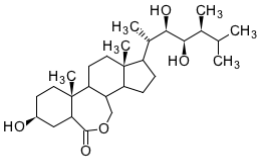
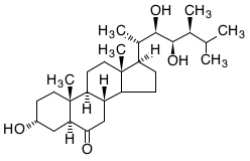
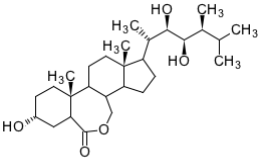
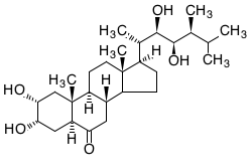
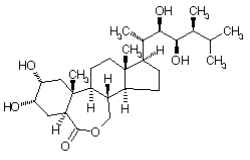
199-229

RP13026	 C02046	 C10860

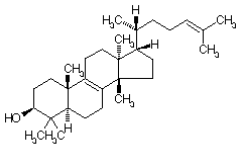
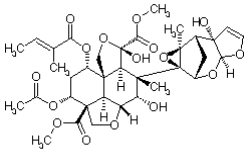
211-242

RP01791	 C00280	 C04676

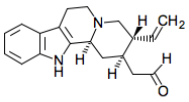
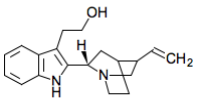
211-243

RP13885	 C15791	 C17733
RP13886	 C15793	 C17735
RP11762	 C15794	 C08814

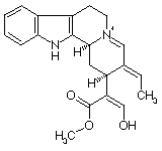
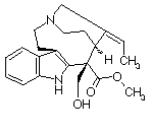
211-426

RP13170	 C08624	 C08748

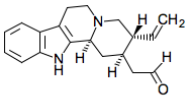
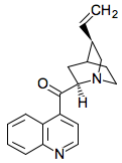
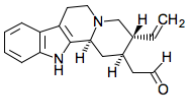
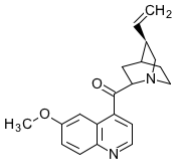
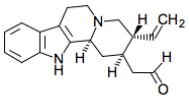
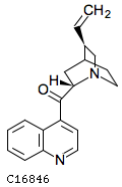
213-222

RP13276	 C16735	 C16727

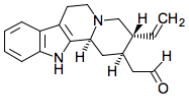
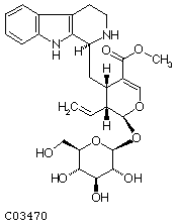
213-246

RP11465	 C03677	 C11691

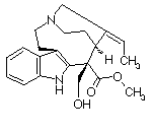
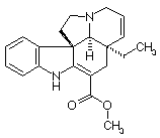
213-263

RP13275	 C16735	 C16726
RP13277	 C16735	 C16728
RP12834	 C16735	 C16846

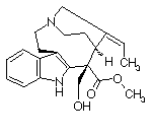
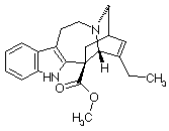
213-391

RP13043	 C16735	 C03470

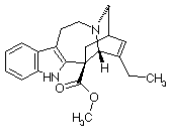
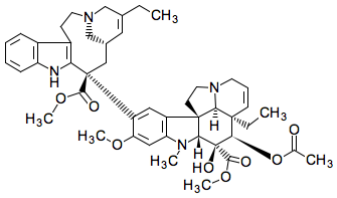
246-277

RP11771	 C11691	 C09244

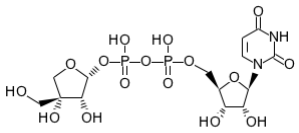
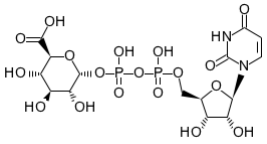
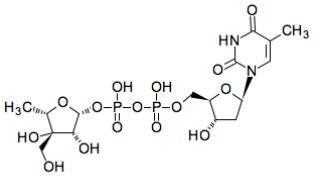
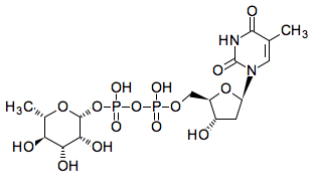
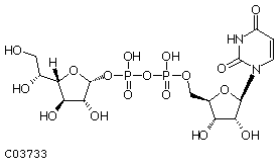
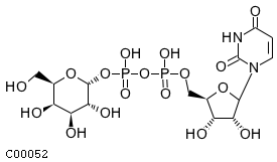
246-278

RP11766	 C11691	 C09107

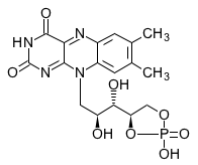
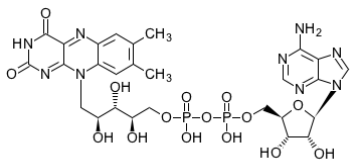
278-499

RP05437	 C09107	 C11641

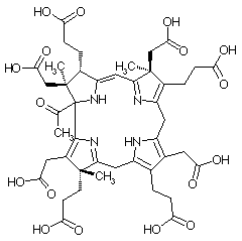
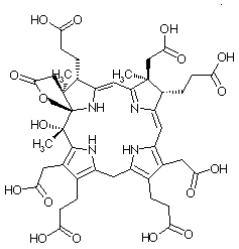
297-317

RP10987	 C01623	 C00167
RP11449	 C03442	 C03319
RP00945	 C03733	 C00052

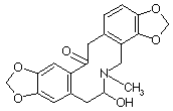
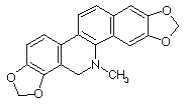
316-492

RP10675	 C16071	 C00016

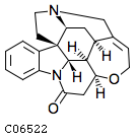
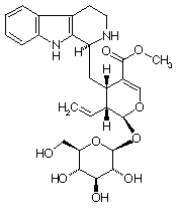
334-401

RP04783	 C06407	 C06406

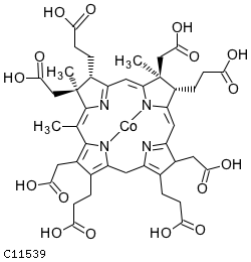
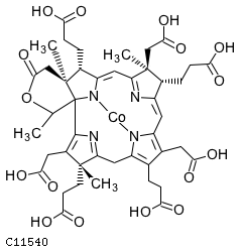
346-361

RP11544	 C05190	 C05191

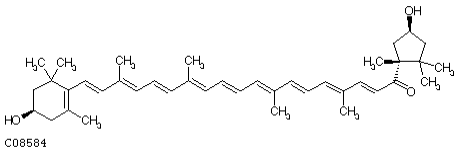
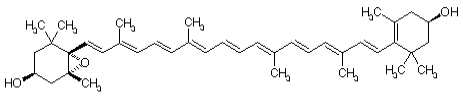
373-391

RP13041	 C06522	 C03470

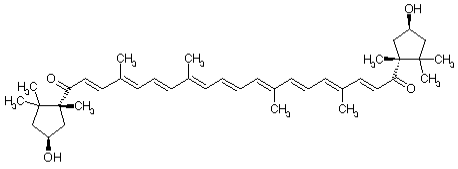
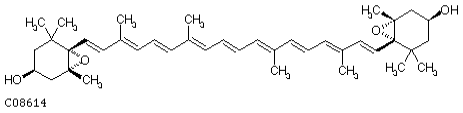
378-427

RP11851	 C11539	 C11540

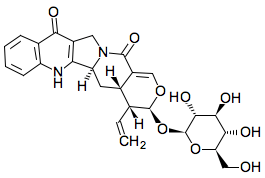
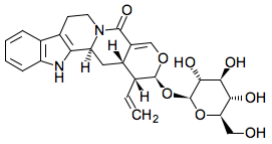
398-424

RP11730	 C08584	 C08579

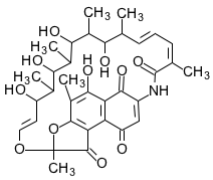
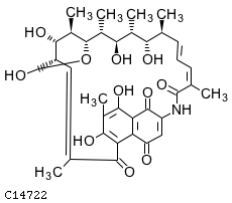
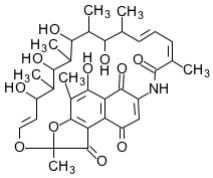
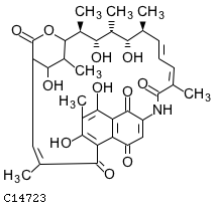
399-438

RP11736	 C08585	 C08614

413-414

RP13273	 C16723	 C16724

420-423

RP10232	 C14724	 C14722
RP10233	 C14724	 C14723

参考文献

Adai AT, Date SV, Wieland S, Marcotte EM. LGL: creating a map of protein function with an algorithm for visualizing very large biological networks. J Mol Biol. 2004 Jun 25; 340(1):179-90.

Bemis GW, Murcko MA. The properties of known drugs. 1. Molecular frameworks. J Med Chem. 1996 Jul 19; 39(15):2887-93.

Cline MS, Smoot M, Cerami E, Kuchinsky A, Landys N, Workman C, Christmas R, Avila-Campilo I, Creech M, Gross B, Hanspers K, Isserlin R, Kelley R, Killcoyne S, Lotia S, Maere S, Morris J, Ono K, Pavlovic V, Pico AR, Vailaya A, Wang PL, Adler A, Conklin BR, Hood L, Kuiper M, Sander C, Schmulevich I, Schwikowski B, Warner GJ, Ideker T, Bader GD. Integration of biological networks and gene expression data using Cytoscape. Nat Protoc. 2007; 2(10):2366-82.

Dalby A, Nourse JG, Hounshell WD, Gushurst AKI, Grier DL, Leland BA, Laufer J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. J Chem Inf Comput Sci. 1992; 32(3):244–255.

De Luca V, St Pierre B. The cell and developmental biology of alkaloid biosynthesis. Trends Plant Sci. 2000 Apr; 5(4):168-73.

Ellis LB, Roe D, Wackett LP. The University of Minnesota Biocatalysis/Biodegradation Database: the first decade. Nucleic Acids Res. 2006 Jan 1; 34(Database issue):D517-21.

Feher M, Schmidt JM. Property distributions: differences between drugs, natural products, and molecules from combinatorial chemistry. J Chem Inf Comput Sci. 2003 Jan-Feb; 43(1):218-27.

Hamada M, Tsuda K, Kudo T, Kin T, Asai K. Mining frequent stem patterns from unaligned RNA sequences. Bioinformatics. 2006 Oct 15; 22(20):2480-7.

Hattori M, Okuno Y, Goto S, Kanehisa M. Heuristics for chemical compound matching. Genome Inform. 2003; 14:144-53.

Hattori M, Yamada T, Oh MA, Goto S, Kanehisa M. Analysis of atom transformation patterns in enzymatic reactions based on the comparison of chemical compound structures(Pathway and database) [in Japanese]. IPSJ SIG Technical Reports 2005; 128:21-25.

Hostettman K, Terreaux C. Search for New Lead Compounds from Higher Plants. CHIMIA International Journal for Chemistry 2000 Nov; 54(11):652-657(6)

Hou BK, Ellis LB, Wackett LP. Encoding microbial metabolic logic: predicting biodegradation. J Ind Microbiol Biotechnol. 2004 Jul; 31(6):261-72.

Huan J, Wang W, Prins J. Efficient mining of frequent subgraphs in the presence of isomorphism. ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining, 2003; 549

Inokuchi A, Washio T, Motoda H. An apriori-based algorithm for mining frequent substructures from graph data. Lecture Notes in Computer Science 2000; 13-23

Inokuchi A, Washio T, Motoda H. Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. Mach. Learn. 2003 Mar; 50(3):321-354

Inokuchi A, Washio T, Nishimura K, Motoda H. A Fast Algorithm for Mining Frequent Connected Subgraphs. IBM Research Report 2002

Kaichi R. Classification of alkaloid by metabolic pathway. Master's Thesis, NAIST 2008

Kanaya S, Kinouchi M, Abe T, Kudo Y, Yamada Y, Nishi T, Mori H, Ikemura T. Analysis of codon usage diversity of bacterial genes with a self-organizing map (SOM): characterization of horizontally transferred genes with emphasis on the E. coli O157 genome. Gene. 2001 Oct 3; 276(1-2):89-99.

Kanehisa M, Goto S. KEGG: kyoto encyclopedia of genes and genomes. Nucleic Acids Res. 2000 Jan 1; 28(1):27-30.

Kohonen T. Self-organized formation of topologically correct feature maps. Biological Cybernetics 1982 Jan; 43(1):59-69.

Kohonen T. Self-Organizing Maps 3rd Ed. Springer 2000

Kotera M, McDonald AG, Boyce S, Tipton KF. Eliciting possible reaction equations and metabolic pathways involving orphan metabolites. J Chem Inf Model. 2008 Dec; 48(12):2335-49.

Kuramochi M, Karypis G. Frequent subgraph discovery. ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining, 2001; 313-320

Langowski J, Long A. Computer systems for the prediction of xenobiotic metabolism. Adv Drug Deliv Rev. 2002 Mar 31; 54(3):407-15.

Lewis RA, Pickett SD, Clark DE. Computer-aided molecular diversity analysis and combinatorial library design. Reviews in computational chemistry, 2000; 16:8-51

Nijssen S, Kok J N. A quickstart in frequent structure mining can make a difference. KDD '04 Proceedings of the thenth ACM SIGKDD international conference on Knowledge discovery and data mining. 2004; 647-652

Oh M, Yamada T, Hattori M, Goto S, Kanehisa M. Systematic analysis of enzyme-catalyzed reaction patterns and prediction of microbial biodegradation pathways. J Chem Inf Model. 2007 Jul-Aug; 47(4):1702-12.

Ray LC, Kirsch RA. Finding Chemical Records by Digital Computers. Science. 1957 Oct 25; 126(3278):814-819.

Raymond JW, Gardiner EJ, Willett P. Rascal: Calculation of graph similarity using maximum common edge subgraphs. The Computer Journal 2002; 45(6):631-644

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski B, Ideker T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 2003 Nov; 13(11):2498-504.

Shinbo Y, Nakamura Y, Altaf-Ul-Amin M, Asahi H, Kurokawa K, Arita M, Saito K, Ohta D, Shibata D, Kanaya S. KNApSAcK: A Comprehensive Species-Metabolite Relationship Database. Biotechnology in Agriculture and Forestry 2006; 57:165-181

Simmond MSJ, Grayer RJ. Plant drug discovery and development. In Chemicals from Plants – Perspectives On Plant Secondary Products; Walton NJ, Brown DE, Eds.; Imperial College Press: London, 1999; 215-249

Takigawa I, Hashimoto K, Shiga M, Kanehisa M, Mamitsuka H. Efficiently finding significant substructural patterns conserved in glycans. JSBi 2008

Talafous J, Sayre LM, Mieyal JJ, Klopman G. META. 2. A dictionary model of mammalian xenobiotic metabolism. J Chem Inf Comput Sci. 1994 Nov-Dec; 34(6):1326-33.

Tohsato Y, Nishimura Y. Metabolic Pathway Alignment Based on Similarity between Chemical Structures. Information and Media Technologies 2008; 3:191-200

Tokimatsu T, Arita M. Viewing Flavonoid Through Metabolic Maps. Saibou Kougaku (in Japanese) 2006; 25:1388-1393

Valler MJ, Green D. Diversity screening versus focussed screening in drug discovery. Drug Discov Today. 2000 Jul; 5(7):286-293.

Verpoorte R. Exploration of nature's chemodiversity: the role of secondary metabolites as leads in drug development. Drug Discov Today. 1998 May 1; 3(5):232-238.

Willett P, Barnard JM, Downs GM. Chemical similarity searching. J Chem Inf Comput Sci. 1998 July 21; 38(6), 983-996.

Xue L, Godden JW, Stahura FL, Bajorath J. Profile scaling increases the similarity search performance of molecular fingerprints containing numerical descriptors and structural keys. J Chem Inf Comput Sci. 2003 Jul-Aug; 43(4):1218-25.

Yan X, Han J. gSpan: Graph-Based Substructure Pattern Mining. Proceedings of ICDM'02: 2nd IEEE International Conference on Data Mining 2002; 721-724.

AntTweakBar http://www.antisphere.com/Wiki/tools:anttweakbar

Ctfile Formats http://www.mdli.com/downloads

Flavonoid Viewer http://www.metabolome.jp/software/FlavonoidViewer

GLUT http://www.opengl.org/resources/libraries/glut/

JME Molecular Editor http://www.molinspiration.com/jme/

MassBank http://www.massbank.jp/

NPEdia http://npd.riken.jp/npedia/

OpenGL http://www.opengl.org/

PerlMol http://www.perlmol.org/

Symyx http://www.symyx.com/downloads/

業績一覧

査読付学術論文

1．**Kenichi Tanaka**, Kensuke Nakamura, Tamio Saito, Hiroyuki Osada, Aki Hirai, Hiroki Takahashi, Shigehiko Kanaya, Md. Altaf-Ul-Amin, "Metabolic pathway prediction based on inclusive relation between cyclic substructures", Plant Biotechnology, 26 巻, 459-468, 2009 年

2．Takashi Oishi, **Ken-ichi Tanaka**, Takuya Hashimoto, Yoko Shinbo, Kanokwan Jumtee, Takeshi Bamba, Eiichiro Fukusaki, Hideyuki Suzuki, Daisuke Shibata, Hiroki Takahashi, Hiroko Asahi, Ken Kurokawa, Yukiko Nakamura, Aki Hirai, Kensuke Nakamura, Md. Altaf-Ul-Amin, Shigehiko Kanaya "An approach to peak detection in GC-MS chromatograms and application of KNApSAcK database in prediction of candidate metabolites", Plant Biotechnology, 26 巻, 167-174, 2009 年

3．Hiroki Takahashi, Kosuke Kai, Yoko Shinbo, **Kenichi Tanaka**, Daisaku Ohta, Taku Oshima, Md. Altaf-Ul-Amin, Ken Kurokawa, Naotake Ogasawara and Shigehiko Kanaya "Metabolomics approach for determining growth-specific metabolites based on Fourier transform ion cyclotron resonance mass spectrometry" Analytical and Bioanalytical Chemistry, 391 巻, 1618-2642, 2008 年

4．Yoko Iijima, Yukiko Nakamura, Yoshiyuki Ogata, **Ken'ichi Tanaka**, Nozomu Sakurai, Kunihiro Suda, Tatsuya Suzuki, Hideyuki Suzuki, Koei Okazaki, Masahiko Kitayama, Shigehiko Kanaya, Koh Aoki, and Daisuke Shibata "Metabolite annotations based on the integration of mass spectral information" The Plant Journal, 54 巻, 949-962, 2008 年

解説記事

1．**田中 健一**, 高橋 弘喜, 平井 晶, 中村 建介, Md. Altaf-Ul-Amin, 金谷 重彦, "代謝反応経路をいかに予測するか？", 日本化学会情報化学部会誌, 27 巻, 2 号, 41-43, 2009 年 8 月

査読付国際会議発表

1．**Kenichi Tanaka**, Kensuke Nakamura, Tamio Saito, Hiroyuki Osada, Hiroki Takahashi, Aki Hirai, Shigehiko Kanaya, Md. Altaf-Ul-Amin, "Pathway prediction focusing on inclusive relation of substructures", The 20th International Conference on Genome Informatics, 2009 年 12 月

２．**Kenichi Tanaka**, Yoko Shinbo, Md. Altaf-Ul-Amin, Aki Hirai, Hideaki Konno, Tamio Saito, Hiroyuki Osada, and Shigehiko Kanaya, "MetClassifier: metabolite classification system based on structural and metabolic pathway information", The 2008 Annual Conference of the Japanese Society for Bioinformatics, 2008 年 12 月

３．Nozomu Sakurai, Yukiko Nakamura, Yoko Iijima, Takeshi Ara, Yoshiyuki Ogata, **Ken-ichi Tanaka**, Koh Aoki, Koei Okazaki, Hideyuki Suzuki, Daisaku Ohta, Shigehiko Kanaya, Kazuki Saito Daisuke Shibata "A software suite for comprehensive detection, annotation and comparison of peaks detected by LC-FTICR-MS", 56th ASMS Conference on Mass Spectrometry, 2008 年 6 月

４．Koh Aoki, Yoko Iijima, Nozomu Sakurai, Yukiko Nakamura, Yoshiyuki Ogata, **Ken-ichi Tanaka**, Kunihiro Suda, Tatsuya Suzuki, Hideyuki Suzuki, Koei Okazaki, Masahiko Kitayama, Shigehiko Kanaya, Daisuke Shibata "Metabolite annotations based on the integration of mass spectral information collected using LC-FTICR-MS", 5th International Conference on Plant Metabolomics, 2008 年 7 月

５．Kaori Chida, Taketo Okada, Takayuki Tohge, Akira Oikawa, **Kenichi Tanaka**, Shigehiko Kanaya, Mami Yamazaki, Kazuki Saito "Non-target analysis of Kampo prescriptions", 5th International Conference on Plant Metabolomics, 2008 年 7 月

国内会議

１．**田中 健一**, 真保 陽子, Md. Altaf-Ul-Amin, 旭 弘子, 黒川 顕, 平井 晶, 有田 正規, 太田 大策, 金谷 重彦 "構造情報に基づいたメタボライト分類支援システムの開発", 第３０回情報化学討論会, 2007 年 11 月

２．**田中 健一**, 真保 陽子, Md. Altaf-Ul-Amin, 旭 弘子, 黒川 顕, 有田 正規, 時松 敏明, 金谷 重彦 "構造情報に基づいたメタボライト分類支援システムの開発", 日本ケミカルバイオロジー研究会第２回年会, 2007 年 5 月

３．千田 かおり, 岡田 岳人, 峠 隆之, 及川 彰, **田中 健一**, 川合 利佳, 南部 羽蘭, 金谷 重彦, 斉藤 和季, 山崎 真巳 "漢方処方のメタボロミクス　―柴胡剤類を中心として―", 日本生薬学会第 55 回年会, 2008 年 9 月

受賞

１．**田中 健一**：第３０回情報化学討論会, ポスター賞, 2007 年 11 月