

Xpress-MPによる困難な混合整数計画問題の解決： MIPLIB 2003でのケーススタディ

リチャード・ランディ(Richard Laundy)、ミハエル・ペレガード(Michael Perregaard)
Fair Isaac, Leamington Spa, Warwicks CV32 5YN, United Kingdom
{richardlaundy@fairisaac.com, michaelperregaard@fairisaac.com}

ガブリエル・タバレス(Gabriel Tavares)、ホリア・ティピ(Horia Tipi)、アルキス・バザコプーロス(Alkis Vazacopoulos)
Fair Isaac, Englewood Cliffs, New Jersey 07632
{gabrieltavares@fairisaac.com, horiatipi@fairisaac.com, alkisvazacopoulos@fairisaac.com}

多項式時間アルゴリズムで、混合整数計画(MIP)問題の求解のためのものは知られていないにもかかわらず、近年、各種の難易度の高いMIPの求解で、目覚ましい進歩が見られた。この論考では、MIPLIB 2003テストセットの難易度の高い問題に注目し、Xpress-MPをどう活用して、以前は求解が困難だとされていた問題の解をどう求めるかを示していく。

キーワード: 混合整数計画(MIP)、MIPLIB 2003、MIP用ソフトウェア

履歴: John Chinneck(前・モデリング、方式、解析分野担当編集委員)によりアクセプト。2007年2月投稿受付、2007年10月改稿、2008年5月アクセプト。Articles in Advanceにてオンライン公開。

1. はじめに

過去10年間で、最先端の混合整数計画(MIP)求解プログラムには、大幅な性能向上が見られた。この性能面での向上は、コンピュータの性能向上ともあいまって、求解可能な問題のサイズと複雑さを劇的に引き上げた。その結果、MIPは今や多くの業界で、意思決定を中心とした応用分野に一般的に使われている。

Xpress-MPは、数理モデルおよび最適化ツールの製品群で、線形、整数、二次、非線形、確率の各計画問題の求解に使用する(Ashford 2007, Dash Optimization 2005, 2008, Gueret et al. 2002)。Xpress-MPは大半のコンピュータプラットフォーム上で、様々なキャパシティとサイズの問題を解くのに利用できる。

Xpress-MPは、PC上での線形計画(LP)モデルのモデル化および求解ツールとして1983年に初めて発売された。その後、1986年に線形計画の分枝限定法実行コードが作られ、MIPモデルの求解のために拡張された(Wolsey 1998を参照)。それ以来、新たな探索法や求解技法を加えていくことで、コードは一層洗練されたものとなった。MIPの性能向上の基本となる変化は次のとおりである。

- ・ 基底となるLPソルバーの、ロバストで高性能な実装。ハードウェアが同一だとしても、20年前の実装より100倍近く高速である
- ・ 根および分枝限定木の両方に適用される、切除平面の自動生成
- ・ 強力な分枝法
- ・ ヒューリスティクス

Xpressに実装されたMIP求解の新技法により、求解時

間が桁違いに短縮されることになった。その結果、以前は解決不能と考えられていた問題が、現在は短時間のうちにごく当たり前に解かれている。新しいアイデアには、特定の問題分野にしか適用できないものがある。しかし、こうしたアイデアを技法のツールボックスに多数揃えて充実させていくことで、実用的な問題から数多く生じた広範囲にわたる問題をXpressが解決できるのである。

Xpress-MPに内蔵されたMIP関連の技法が大幅に充実してきた結果、われわれは、MIPLIB 2003にある困難な問題のいくつかに立ち戻り、何が達成できるかを確認する時期だと判断した。Xpressに実装された一連の技法がどう活かされて、かつては解決が困難だと思われていたいくつかの問題を最終的にどう解決するかを示していく。

2. MIPLIB 2003

MIPLIB 2003は、広く利用されている標準的なベンチマークで、各種MIPアルゴリズムの性能を比較するものである。

ライブラリは<http://miplib.zib.de>で公開されている(Achterberg et al. 2006b)。60の(最小化)問題で構成されており、それぞれの問題ごとに特徴を持っている。詳細はAchterbergらによる文献(2006a)に記述されている。

MIPLIBテストセットにある問題は、さまざまな分野での問題に由来したもので、非常に簡単なものから、LP緩和問題でさえ求解が困難であるようなものまで、広範囲にわたっている。MIPLIBのウェブサイトによれば(2006年10月時点での情報)、MIPLIB 2003の問題は、難しさの度合いで次の3グループに分類されていた。

- ・ 28の問題は、市販のソルバーで1時間以内に解決される
- ・ 18の問題は、最適解が知られてはいるが、前述の「1時間以内」の条件を満たしていない
- ・ 14の問題が未解決である

調査を開始した時点で、問題の1つである stp3d に対する既知の解法はまだ存在せず、また、未解決に分類されていた14の問題のうち、11問が依然未解決のままであった。ちょうどその頃、3つの問題に対する解が Ferris (2006) によって発見されたばかりであった。Ferris は Condor での GAMS グリッドコンピューティングを使用して問題 a1c1s1 および timtab2 の解を報告したほか、巡回セールスマン問題である swath がサブツアー消去カットの適用により迅速に求解できることを示した。

本稿の投稿に先立つ約1年間で、次のとおり、MIPLIB での未解決問題の数は17から8に減少した。

2005年12月、Balas および Saxena (2005) により、問題の分割による閉包を使用することで arki001 の解が最適であることが証明された。

2006年6月、MIPLIB のウェブサイトによれば、問題 glass4 および roll3000 が最新の MIP ソルバーにより解決された。

2006年9月、a1c1s1、timtab2 および swath の最適値が Condor の GAMS グリッドコンピューティングにより発見された (Ferris 2006)。

2006年11月、Vazacopoulos ら (2006) により、Xpress 2006B を使用して atlanta-ip、msc98-ip、rd-rplusc-21 の最適値が発見された。

本稿では Xpress-MP1 を用いて、Vazacopoulos ら (2006) により提示された最適解の発見につながる最適化戦略のいくつかと、従来未解決であった問題 profold および sp97ar の最適解発見のための戦略 (§5 を参照) を取り扱っていく。また、過去に解決された5つの問題に対し、(最適性までの) 求解時間を改善したことを示し (§4 を参照)、残る6つの未解決問題すべてに対し、改善された解法を提示する (§7 を参照)。

困難な MIPLIB 問題の検討に入る前に、最初に簡単な問題にいくつか注目する。1時間以内で求解可能な MIPLIB の問題群に関する Xpress-MP の性能を、セクション3で検証する。

3. 1時間以内で求解可能な問題

表1には、MIPLIB 2003からの問題が29挙がっている。Xpress 2006B がデフォルト設定を使い、最適解を1時間以内で求められる問題である。すべてのテストは、Xeon 3.0GHz 64ビットデュアル、4GB RAM、Windows XP 上で、Xpress をシリアルモードで走行させて行った。

¹デフォルト設定では、MIP の最良解が最適性の0.01%以内にあると、Xpress は動作を停止する。こ

の実験では、最適性が証明されるまで木探索を実施した。

表1にある結果で最も興味深い点は、次のとおりである。

- ・ 6つの問題は根ノードで求解できる
- ・ 9つの問題は、Xpress 2006B では1秒未満で解決される
- ・ 比較的簡単な問題群に対する Xpress 2006B の平均計算時間は、164.6秒である。1秒以内に解決された問題を除外した場合、残る20問での平均求解時間は265秒である
- ・ 比較的簡単な問題のうちでも、難易度の高い問題は mas74、mzzv11、pk1 の3つであり、Xpress 2006B での求解時間はそれぞれ3,332秒、359秒、228秒である

3.1. Xpress の性能に関する系譜の概略

ここで、過去数年間の Xpress の性能向上ぶりに着目し、その理由をいくつか述べていく。

図1は、30分以内に解決できた MIPLIB 2003 の問題の数を示したものである。求解は、年1回ペースでリリースされる Xpress-MP の過去4つの版により行った。同一のコンピュータ (Xeon 3.0GHz デュアル、64ビット、Windows XP) を使用する場合、Xpress 2003G では22問のみであるが、Xpress 2006B ではさらに6つの問題を1問あたり30分の制限時間内に解決する。

表2に、Xpress の新しい方のリリース (2004D、2005B、2006B) について、Xpress 2003G と対比させ、MIPLIB 2003 の比較的簡単な問題群に対するスピードアップ要因を示している。4つのリリースすべてで1秒以内に求解できる問題 (fixnet6、gesa2、gesa2-o、modg1ob、modg1ob、p2756、pp08a) は除外した。

Xpress の性能向上は、1つの改良に帰するものではなく、数多くのアルゴリズムの改良によるものであり、組み合わせられて技法のツールボックスを形成している。この技法のうちどれか1つが特定のモデルに適用できるのであれば、そのモデルの求解時間を劇的に削減することができる。アルゴリズムの改善点のいくつかを以下に述べる。

3.1.1. プレスルブ MIPLIB 問題の大半はプレスルブ実施によるメリットがある。プレスルブの実施技法は誕生して久しく (Brearley et al. 1975 を参照)、計算時間の点からもそれほど労力のかからない技法でありながら、問題のサイズを縮小し、MIP の定式化を強固なものにする。問題のサイズを縮小することのメリットは、木探索における各ノードの求解時に発揮され、通常は線形的なスピードアップが得られる。いずれにしろ、係数の絞り込み (Savelsbergh 1994 を参照) のような技法を使った定式化の強化は、探索木サイズの縮小につながり、より大幅なスピードアップという結果が得られる。Xpress で現在利用できるプレスルブ技法の数については、われわれがモデルの定式化の解析を行い、強化方法の発見が進むにつれて増加してきた。以前は「悪い」定式化

¹ コンピュータによるこの実験では、リリース2006Bのバージョン17.10.04を使用した。

表1. MIPLIB 2003問題の分枝限定(B&B)ノード数および求解時間(デフォルト設定のXpress-MP 2006Bにより1時間未満で最適性まで解決されるもの)

問題	Xpress B&Bノード数		Xpress所要時間†		最適値
	解まで	最適性まで	解まで(秒)	最適性まで(秒)	
10teams	26	51	2	2	924
aflow30a	3,416	7,023	23	43	1,158
air04	166	185	36	36	56,137
air05	209	261	31	32	26,374
cap6000	1,389	1,937	7	9	-2,451,377
disctom	1	1	4	4	-5,000
fiber	56	69	<1	<1	405,935.18
fixnet6	9	11	<1	<1	3,983
gesa2	1	1	<1	<1	25,779,856.4
gesa2-o	1	1	<1	<1	25,779,856.4
harp2	30,942	67,239	90	149	-73,899,798.84
manna81	1	1	<1	<1	-13,164
mas74	46,709	2,380,685	35	3,332	11,801.1857
mas76	1	219,797	<1	159	40,005.0541
misc07	2,220	21,359	8	65	2,810
mod011	1,001	1,277	64	72	-54,558,535
modglob	1	11	<1	<1	20,740,508.1
mzzv11	1,047	1,145	345	359	-21,718
mzzv42z	121	125	48	48	-20,540
nw04	28	161	15	17	16,862
opt1217	1	1	<1	<1	-16
p2756	8	15	<1	<1	3,124
pk1	157,892	232,391	149	228	11
pp08a	124	213	<1	1	7,350
pp08aCUTS	88	181	<1	1	7,350
qiu	6,245	10,383	120	147	-132.873137
rout	4,165	10,323	34	67	1,077.56
set1ch	1	1	<1	<1	54,537.75
vpm2	19	1,341	<1	2	13.75

†データはデュアルXeon 3.0GHz、64ビット、4GB RAM、Windows XP環境で取得した。

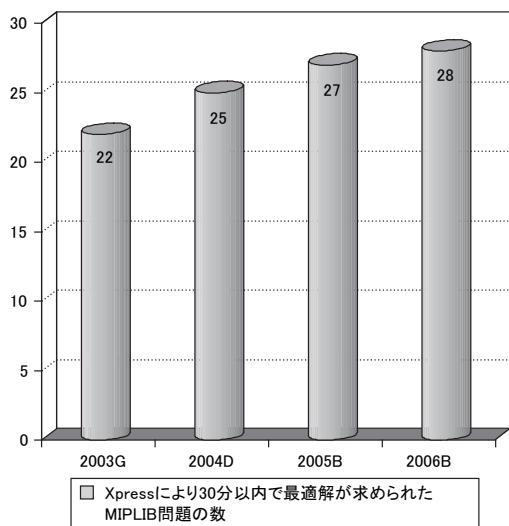


図1. 過去4年間でリリース全般において、(Pentium 4 3.6GHz、Windows XPで)デフォルト設定を使ったXpressにより、30分以内で最適解が求められたMIPLIB 2003問題の数

3.1.2. 切除平面 切除平面の生成により、求解時間に大きく影響を与えることができる。いくつかの問題の場合、LPの求解に分枝法はほとんど効果がなく、そのため、探索木のサイズが指数関数的に増大する。これは、LPが退化していて、同じ目的値を持った数多くのノードが作り出せる場合に特有の問題である。MIPLIB 2003の10teams、manna81、opt1217モデルは、典型的な例である。これらの問題に対して優れた切除が生成され、インテグラリティ・ギャップを縮めることができれば、残っている退化したLP解の中からMIPの最適解を見つけるといふ問題となる。

表2 MIPLIB 2003の比較的簡単な問題における、Xpressソフトウェアリリースの2003年版から2006年版までの性能比較

問題	求解時間 [†] 2003G (14.27) (s)	2003Gに対する最適性求解 の高速化比率		
		2004D (15.30.12)	2005B (16.10.09)	2006B (17.10.04)
10teams	451	30.1×	32.2×	225.5 ×
aflow30a	141	0.8×	0.8×	3.3 ×
air04	75	3.1×	3.8 ×	2.1×
air05	54	1.5×	1.2×	1.7 ×
cap6000	46	2.0×	2.2×	4.6 ×
disctom	≥ 20,000	≥ 6,670 ×	≥ 6,670 ×	≥ 5,000×
fiber	3	3.0×	3.0 ×	3.0 ×
harp2	437	1.4×	1.5×	2.7 ×
manna81	≥ 6,000	≥ 1,500×	≥ 6,000 ×	≥ 6,000 ×
mas74	3,781	1.2×	1.3 ×	1.1×
mas76	112	1.1×	0.9×	0.7×
misc07	98	1.1×	0.9×	1.3 ×
mod011	129	1.4×	1.4×	1.6 ×
mzzv11	97,889	< 1.5×	119.7 ×	113.2×
mzzv42z	≥ 20,000	n/a	≥ 415 ×	≥ 385×
nw04	52	4.7 ×	4.0×	2.7×
opt1217	≥ 35,000	n/a	n/a	≥ 35,000 ×
pk1	139	0.6×	0.7×	0.6×
pp08aCUTS	2	2.0 ×	2.0 ×	2.0 ×
qiu	196	1.3 ×	1.3 ×	1.2×
rout	86	0.3×	0.9×	1.2 ×
set1ch	2,965	0.8×	2,965 ×	2,965 ×
vpm2	6	2.0×	2.0×	3.0 ×

注. 最も速い求解時間を太字で示している。

[†]データはデュアルXeon 3.0GHz、64ビット、4GB RAM、Windows XP環境で取得した。

Xpress内部で生成される切除は、次のように分類できる。

- ・ゴモリー／lift-and-project(離接)カット
- ・クリークカット
- ・lifted coverカット
- ・MIR(混合整数丸め)カット
- ・インプリケーションカット
- ・フローパスカット

ゴモリーカット(Gomory 1960)は、汎用的なカットで、非整数またはバイナリ変数の完全性を用いて最適な単体表から生成される。lift-and-projectカット(Balas et al. 1993, Balas and Perregaard 2002, Cornuejols 2007, Perregaard 2003)は離接カットであり、必ずしも実行可能でなく最適でもない単体表から生成されたゴモリーカットと等価であると示せるものである。こうしたカットは、MIPLIB問題の大半で生成でき、通常はインテグラリティ・ギャップを縮めるのに大いに貢献するものである。

残るカットは行列内のさまざまな構造に作用するか、ある属性の発見に頼るものである。クリークカット(Atamturk et al. 1998を参照)は、バイナリ変数の集合の和が1未満に制約されるカットである。基準を満たさないクリークカットは、モデルの他のクリークより生成できる。これらは通常、MIPLIB 2003でのair04、air05、nw04モデルのような、航空機の乗務員のスケジューリング問題において生成される。

lifted coverカット(Crowder et al. 1983, Van Roy and

Wolsey 1987)は、条件を満たさないカバーカットが生成できるような、行列内部でのナップサック制約から生成される。このカバーカットはリフティングされて、ナップサックにある他の変数が含まれる。ただし既にカバーにあるものは含まない。Xpressは、移動された一般化上界値(GUB)カバーカットなど、lifted coverカットの拡張であるさまざまなカットを生成する。これは、ナップサックに入った変数がクリーク不等式(またはGUB)の要素であるようなナップサック制約から生成されたlifted coverカットである。cap6000 モデルは、リフティングされたGUBカバーカットが生成できる典型的な問題である。

MIRカット(Wolsey 1998を参照)は、本質的にはゴモリーカットで、元の問題の制約か、またはその単純な集合体より生成されたものである。lifted coverカットのように、MIRカットは係数が非一様である制約からのみ生成できる。MIPLIB 2003問題の多くは、ナップサックタイプの制約か、あるいは集約されてナップサック制約を生み出せるような制約を含んでいる。

インプリケーションカット(Hoffman and Padberg 1991)は、あるバイナリ変数が他の変数の値域を包含する場合に生成可能である。一例として、可変上界値(VUB)は、元のモデルにVUB制約の集合体が含まれる場合に生成できる。このカットは、疎(スパース)である性質のため、非常に効果的である。

フローパスカット(Padberg et al. 1985)は、モデルが、非ゼロのフローが固定金額を負担しているようなネットワーク構造を有していれば生成できる。フローパスカットは、非常に効果的な手法となりうる。たとえば、set1chモデルが現在1秒未満で解決される理由は、フローパスカットが関わったことにある。

Cordierら(1999)は、Xpressに内蔵されたカットのいくつかについて、より詳しく記述している。

3.1.3. 分枝変数の選択 分枝変数の選択を行うことで、木探索のサイズに対して大きな効果が得られる。通常は、目的関数の変化または構造的な変化という点で、問題に大きな影響のある分枝変数を選択することが最良である。分枝変数選択の改善による木サイズの縮小効果は、よりよい分枝変数の発見に時間を費やすコストに見合うほどに十分大きくなりうる。強力な分枝法(Applegate et al. 1995)とは、分枝候補に双対の代入を実行して、候補での分枝の効果を確立させるという先取的な方式である。この方式は労力に比して効果が得られない可能性があり、強力な分枝法はデフォルト設定では限度が設けられている。航空会社のスケジューリング問題であるair04、air05、nw04は、強力な分枝法のメリットがある典型的な問題である。

3.1.4. ノード前処理 求解の前に木ノードで前処理を実行することで、木探索のサイズを小さくできる。これにはコスト固定法と、限界値の絞り込みが含まれる。MIPLIB 2003のテストセットの中では、cap6000モデルがノード前処理のメリットのあるモデルの1つである。このモデルに対しては、ノードを探索しきるまで木探索を2,000回以上行えるが、分枝の大半は、ノード前処理で回避することができる。

3.1.5. ヒューリスティクス いくつかの問題にとつては、木探索でMIPの解を求めることは困難である。たとえば、MIPLIB 2003問題のdisctomは、非常に退化したもので、整数でない変数で分枝を行っても、その大半がほぼ同数の非整数変数と、全く同じ目的関数値を持った子問題が生成される結果に終わる。木探索を深いレベルまで行ってはじめて実行不可能性が検出され、整数解を求めることは非常に困難である。しかし、丸めヒューリスティクスにより最適解を発見することは相当に易しい。トップノードでは最適ギャップがなく、問題は数秒で解決されるためである。

3.2. 困難な問題の求解戦略

簡単なMIPLIB問題では、ここまで述べてきた技法が非常に効果的であることを概観してきた。難易度の高いMIPLIB問題に対しては、木探索の指数関数的な増大を防ぐことがより大きな課題である。たとえば、木探索のサイズが増大し続けるのであれば、Xpressを1カ月間実行させても大して有用ではないだろう。ここで直面している問題の例として、ノードを探索し尽くすまでに20段階掘り下げることが必要な木探索について考えてみる。探索木のサイズは100万ノード単位となり、仮に各ノードの解決に1秒かかるのであれば、問題の求解には最終的にまる2週間弱を要することになる。では、ノード探索の深さが40に増えたケースを考えてみよう。探索木のサイズは $1e+12$ 単位のノードとなり、たとえ個々の解決が100分の1秒であっても、全体の求解時間はおよそ350年になる。したがって、木探索のサイズは簡単に手に負えないものとなりうる。また問題の求解を試みても、その可能性があるかどうかはすぐに明らかとなる。

追跡不能に思われる問題の求解を試みるとき、どんな選択肢があるだろうか？最初に検討すべきことは、定式化である。定式化が劣悪だと、求解がはるかに困難なモデルが出来上がることもある。プレスルブによる自動的な定式化の改善も、必ずしも可能な話ではない。MIPモデルの定式化は職人技のようなところもあり、定式化を改善していく方法について一般的なアドバイスを行うのは難しい。

ある1つのモデルを、求解の簡単な小さい問題に分解できるかを検討する価値はある。時として、モデルが主要な決定変数の集合を含むことがある。それらの変数がいったん確定すれば、いくつかの小規模なモデルに分かれるようなモデルの場合である。小規模なモデルのそれぞれは、求解が割合簡単かもしれないが、組み合わせられたモデルでは格段に難しい。そのときのモデル全体の求解戦略は、主要な決定変数について考えるすべての組み合わせを残らず挙げて、小規模な子問題を解いていくことである。

定式化が改善できなければ、木探索に役立つように、さまざまな戦略を試してみなければならぬ。木探索の早い段階で優れた整数解を見つけることは、探索の刈り込みに資する機会が多く、分枝法の改善にも役立てることができる。極端な難問には、最初の求解で非常に良好な解の発見を試みた後、次の求解において、その解から得られたカットオフ値

を使い、最適性の証明を試行するだけの価値がある。2回の実行での設定は、まったく異なったものになることもある。

簡単な部類のMIPLIB問題に対し、切除戦略の重要性を検討してきた。適切なカットを見つけ出すことで、探索木のサイズを飛躍的に縮小できる。とはいえ、カットを追加しすぎると、行列の探索を停滞させてノードの最適化を遅らせる可能性がある。デフォルトでの切除戦略は、困難な問題に対しては慎重すぎる面もある。よって、問題に加えるカットの数を増やしてみる価値はある。

木探索が完了する見込みがありそうならば、全数探索アプローチに持ち込むことがいつでも可能である。強力な分枝に一層注力するか、あるいは木ノードの探索順序を変更することで、木探索のサイズの縮小を試すことができる。Xpressの並列走行もまた有用となりうる。並列処理プログラムでは木探索を別々に実施し、以前にLaundy (1999) による文献に記述されて出回っていた並列処理コードの動作と類似したやり方で、異なるスレッドの探索を進め、多くの場合に線形的なスピードアップをもたらす。

4. 近年解決された問題

ここで、Vazacopoulosら (2006) 以前に解決されたうち、最後に立証された5つの問題に使用して、最適性まで求解を行った戦略について述べていく。具体的にはa1c1s1、arki001、glass4、roll3000、swathの5問である。なおMIPの1つ (a1c1s1) は、1台のコンピュータで解決されるのは初めてであり、他の1つ (swath) は、1台のコンピュータ上で初めてオリジナルの形のまま解決されることを注記しておく。

Xpressのパラメータに加えた変更、という点から戦略について述べていく (パラメータの完全な記述についてはXpress-MP Optimizer リファレンスマニュアル (Dash Optimization 2005) を参照)。

4.1. a1c1s1

問題a1c1s1は、Van VyveおよびPochet (2001) が最初に検討した、ロットのサイズ決定問題である。この問題の場合、当初のインテグラリティ・ギャップの大部分は、切除法によって縮小できる。しかしギャップの最後の10%については、カットの追加でも分枝でも、縮めることが非常に難しい。この理由により、これに代わる戦略が必要とされた。

われわれが取った求解のアプローチには、問題を簡単な問題の集合へと分解することが入っている。まず、全体の強力分枝と一緒に部分的な分枝限定を実行し、第1段階で128の子問題を作成した。30分以内に解決できなかった子問題については、再びそれぞれを5つの子問題へと分割し、すべての子問題が解決されるまでこれを繰り返した。

テスト用のコンピュータには、3GHzのIntel Core Duo 2、4GB RAMのものを使用した。実験では、mipthreads=2の設定でXpressの並列処理を用いた。最適性の証明に要した時間は約32時間であった。

こうした方法を適用して作られた子問題の合計は230で、そのうち212が「最終」の子問題、つまり、分割ルールが適用されない問題であった。これらの子問題の作成に要した時間は約3,800秒、求解に要した時間は概ね11万秒で、そのうち7万2,000秒が「最終」子問題にかかったものであった。この手順で解決されたノードの合計数はおよそ210万であった。

子問題の求解のため、いくつかのXpressの制御設定を使用して、最適化実施性能を一段と高めた。端的に述べると、すべての未解決ノードから、最良のLP分枝を持ったノードを選ぶノード選択ルールを使用し (nodeselection=2)、Xpressでのカットと、強力な分枝法の実行にさらに注力した (cutstrategy=3、covercuts = 50、gomcuts =10、cutfreq= 2、cutdepth = 20、treegomcuts=0、sbeffort=2)。また、カットオフ値は11,534を使用した。

Ferris (2006) も、近年alc1s1の解決を報告した。そのために、FerrisはCondorシステムにGAMSグリッドコンピューティングを使用し、CPU時間で3,452時間を費やした。われわれのCPU時間は約58時間であるから (使用したのも2CPUである)、Xpress 2006Bを使用するわれわれのアプローチは、Ferris (2006) によるものよりもCPU時間が約60分の1である。

問題alc1s1の最適な目的値は、11,503.444125である。

4.2. arki001

問題arki001は、冶金産業で生じた非線形問題の緩和問題である。この問題の良解を見つけることは可能なのだが、発見された最良解が実際に最適であることは、2006年にBalasおよびSaxena (2005, 2008) が分割閉包に由来したカットを使用することで求解するまで証明されなかった。BalasおよびSaxena (2005) の報告での求解時間は約65時間で、そのうち54時間がランク1の分割切除の生成に費やされ、11時間が強化されたMIPの求解に使われた。

Achterbergら (2006b) は、この問題を9時間弱で解決した。また、Achterberg (2006) はSCIPと強力な分枝法およびコンフリクト解析の解法を積極的に使用して、5.2時間で約180万ノードを計算して問題を解決した。

Vazacopoulosら (2006) は、Xpress 2006Aによりarki001を4時間で求解したと報告している。Xpress 2006Bは、デュアルXeon 3.0GHzで (1スレッドを使い) 285万ノードを計算して、この問題を3.9時間で解決する。この結果を達成するために、(lift-and-projectカットを含め) ルートノードと分枝限定の両方で積極的な切除生成を使用する必要と、強力な分枝に費やされる労力を削減する必要がある (cutstrategy = 3、covercuts = 5、gomcuts =10、lnpbest=150、cutfreq=1、treegomcuts=4、heurstrategy=0、varselection=3、sbeffort=0.1)。

問題arki001の最適な目的値は7,580,813.046である。

4.3. glass4

問題glass4は、Luzzi (2002) により最初に検討されたネスト化の問題である。この問題は、カットオフ値を使えばXpressでの求解はずっと簡単である。というも、それによってコスト固定化の削減が実施可能で、カットオフ値よりも大きな境界を持ったすべてのノードで探索を完了できるためである。したがって、第1段階ではもっぱら良解の発見を行い、第2段階で第1段階のカットオフ値を用いた求解に専念するという2段階のプロセスを使用して解を求めた。

glass4の優良解を求めるのが困難な理由の1つは、1つの変数のコスト係数が $1e+6$ となっていて、値が1か2である他のすべてのコスト係数よりも格段に大きく、それに目的関数が大きく影響されることである。これが、最後に発見した解よりもわずかに優れていないものを求めるのにも、木探索に時間がかかりすぎる原因となる。この事象を回避するため、mipaddcutoffの設定を行った。この制御設定は、発見されたMIPの解の値に付加され、新たなカットオフ値を与えるものである。その結果、最後に発見した解よりも少なくともmipaddcutoffの値以上に良好ではない境界を持ったあらゆるノードが除外される。多くの場合、解同士の差異が $1e+8$ ほどあるため、mipaddcutoffの値には $-9.98e+7$ を設定した。また、探索が最良の予想解が存在しそうなノードへバックトラックするように、木探索にバックトラック戦略を設定すること (backtrack = 1)、そして、深く探索する際に両方の子ノードが解決されるように分枝の選択を設定すること (branchchoice = 1)、この両者によって求解の可能性を高めた。これらの設定を使い、3GHzのIntel Core 2 Duoプロセッサで90秒間に14の解が求められた。最後に求めた値の1,200,012,600が、この問題の解であると判明した。

既知の最良解 (1,200,012,600) から算出したカットオフ値を使い、Xpress 2006Bは、3GHzのIntel Core 2 Duoプロセッサで50万ノード近くを計算し、この値が実際に最適解であることを9分間で証明している。

問題glass4の最適な目的値は1,200,012,600である。

4.4. roll3000

問題roll3000は、Kroon (2002) により最初に検討された車両運用の問題である。

Xpress 2006Aは、この問題の最適性まで求解できた最初のソルバーの1つである (Achterberg et al. 2006b, Vazacopoulos et al. 2006を参照)。ノード選択戦略を改良して、最良下界値を持ったノードを選ぶ (nodeselection = 2)、切除法に一層の重点を置く (特に、“追加”ゴモリーカットの生成において) という特別な設定が使用された。Xpress 2006Aは、3.0GHzのデュアルXeonで、約280万ノードを15.5時間かけて最適性の証明を行った。

さらに特別な設定を使うと、この所要時間はわずか13分にまで削減できる。そのためには、強力な分枝法の実施に一段と注力することと、探索木のトップノードでのゴモリーカットおよびlift-and-projectカットの生成を増やすことが必要である。特定の設定 (gomcuts = 20, lnpbest=100, lnpiterlimit = 50, cutfreq = 1, treecovercuts = 2, treegomcuts = 2, heurstrategy = 3, sbefort = 4) を使えば、Xpress 2006Bではおよそ10万ノードを計算して、この問題の最適性を証明する。

Ferris (2006) は、CondorのGAMSグリッドコンピューティングを使用して、roll3000はCPU時間にして約50時間で解決されたと述べている。Achterberg (2006) は、SCIPと強力な分枝法および適切なコンフリクト解析を使って、3.2GHzのPentium 4コンピュータで410万ノードを計算し、この問題を10.3時間で求解した。

問題roll3000の最適な目的値は12,890である。

4.5. swath

問題swathは、合成開口レーダー調査用の作業計画モデルである。このモデルは航空機の経路を最適化するもので、巡回セールスマン問題 (TSP) に類似している。

問題固有のカットが加えられれば、TSPの求解ははるかに容易である。近年、Ferris (2006) は5度にわたりサブツアー消去カットを追加してから、swathを解決した。これが結果的に32の追加制約となっている。この“拡大された”問題は、1台のマシンと標準的なMIPソルバーを使って20分未満で解かれた。

しかしこの問題は、問題alc1s1の求解に使用したものの (§4.1を参照) と同一の最適化手順を使用することで、問題固有のカットを使わずに求解できる。Intel Core Duo 2コンピュータの両方のコアを使い、問題swathの最適性の求解に要した時間は、約66時間であった。

われわれの方法を適用して作成された子問題の合計数は516で、うち422が最終の子問題であった。これらの子問題の作成に要した時間は900秒、子問題の求解時間はおよそ23万4,000秒、そのうち7万9,000秒が「最終」子問題に対応するものであった。この手順で求解したノードの合計数は約6,000万ノードであっ

た。

このケースでは、いくつかのXpress制御変数を再度使用して、より高い最適化性能が得られた。すべての未解決ノードから最良下界値を持ったノードを選ぶノード選択ルールを使用し (nodeselection = 2)、Xpressでのカット (端的には、lift-and-projectカット) と強力な分枝法の実行にさらに注力した (cutstrategy = 3, covercuts = 50, gomcuts = 20, cutfreq=5, cutdepth=50, treegomcuts=0, sbefort = 5, sbiterlimit = 100)。また、カットオフ値は468を使用した。

CondorシステムのGAMSグリッド (Ferris 2006を参照) では、CPU時間で3万6,000時間以上かけてもオリジナルの (変形していない) 問題swathの最適性を証明できなかったことを注記しておく。

問題swathの最適な目的値は467.407491である (Ferris 2006)。

5. 初めて解決された問題

ここで、2006年9月以前には解決されていなかった、MIPLIB 2003のより難易度の高い問題に目を向けることとする。

MIPLIB 2003の3つのMIP問題であるatlanta-ip、msc98-ip、rd-rplusc-21は、Vazacopoulosら (2006) によりXpress 2006Bを使って初めて解決された。

このセクションでは、Vazacopoulosら (2006) が使用して上記問題の最適性の証明に至った、アルゴリズム面からの各種アプローチについて述べていく。さらに、2つの問題protfold、sp97arの最適な目的関数値は、初めて示されるものである。

結果の概要を、計算時間および対応する最適値を含めて表3に示す。

以下の各項では、問題の求解に使用した個別のアプローチについて記述する。

5.1. atlanta-ip

問題atlanta-ipは、辺の制約がある最小コストネットワーク問題である。インテグリティ・ギャップはかなり小さいが、完全になくすことは非常に困難である。

表3. (以前は未解決だった) 4つのMIPLIB 2003問題の目的値およびXpress-MP 2006Bを使用して最適解の発見に要した計算時間

問題	コンピュータ	求解時間	最適な目的値
atlanta-ip	3 GHz Intel Core 2 Duo	10 hours	90.00987861
msc98-ip	3 GHz Intel Core 2 Duo	1.5 hours	19,839,497.005874
sp97ar	3 GHz Intel Core 2 Duo	>1 month	660,705,645.5
protfold	Xeon 3.0 GHz, 2 CPU, 4 GB of RAM	9 days	- 31
rd-rplusc-21	Xeon 3.0 GHz, 2 CPU, 4 GB of RAM	3.6 hours	165,395.2753

最初にモデルを解析したところ、この問題にはすべての MIP 解法で切除であるように思われ、かつ冗長に見えるような数の制約が含まれることが示唆された。モデルにカットを追加することは、時として求解がより一層困難になる可能性があり、そのため、カットである疑いのあったすべての制約を外した。

次に、目的関数が 2 段階の構造を持つことを発見した。1 つのバイナリ変数の集合は、コストが 1 以上の整数値であり、残りのバイナリ変数は、コストが 10⁻⁶ から 10⁻⁴ の範囲にある。小さい方のコスト係数を除外すれば、簡単な問題が作れる。というのは、残るコスト係数の最大公約数が 1 であることを利用して、カットオフを改善して木探索のスピードを上げられるからである。この簡単にした問題の解は、元の問題に対する実行可能解となる。また、小さい方のコスト係数の合計値は 0.5 未満であるため、簡単にした問題の最適解の値を使い、その変数だけが大きい方のコストに参与しているような目的のカットを、元の問題に追加できる。しかも、簡単な方の問題は、(3GHz Intel Core 2 Duo マシンで) およそ 4 時間、9 万 5,000 ノードで最適性まで求解できる。小さい方のコスト係数による、発見した解での目的関数への貢献分は、0.012 であった。よって、この方法で求めた解は、元の問題の最適解の値から 0.012 以内に位置している。

最終段階は、(各種カットを除外して) 追加した目的のカットによる元の問題の求解である。これには探索木内の重点的な切除と強力な分枝法によって (gomcuts = 0, cutfreq = 1, cutdepth = 20, treegomcuts=0, sbefort=5, sbiterlimit = 500)、4.5 時間、6 万 6,000 ノードを要した。この最終の解を元の問題に戻すと、われわれがカットであると推定した制約値は、この解では実際に冗長であった。

問題 atlanta-ip の最適な目的値は 90.0098786144 である (Vazacopoulos et al. 2006)。

5.2. msc98-ip

問題 msc98-ip は、atlanta-ip に似た (§ 5.1 を参照) 最小コストネットワーク問題であるが、こちらは国土全体の通信ネットワークの設計から生じたものである。

この最適解を求めるため、atlanta-ip の求解に用いたものと同じアプローチを取った。問題 msc98-ip は、atlanta-ip よりも求解がいくぶん簡単である。

Xpress 2006B は、(3GHz の Intel Core 2 Duo マシンで) 第 2 段階で約 8 分の計算時間を必要とする。その結果、最適性に近似する実行可能解が求められる。

最終段階では、Xpress 2006B は同じコンピュータで約 1 時間の計算時間を必要とする。(5.1 と同様に) 重点的な木の切除と強力な分枝法を使用して、4,000 弱のノードを計算する。

問題 msc98-ip の最適な目的値は 19,839,497.005874 である (Vazacopoulos et al. 2006)。

5.3. protfold

問題 protfold はタンパク質の折りたたみモデルである。モデルの退化が、最適性の求解をとりわけ困難にしている。質の高くない解を求めることは簡単であるが、より優れた解を見つけるのは一層難しくなる。

問題の求解を数時間実行させたところ、アクティブな木ノードの数はさほど急速に増えないことが分かった。よって、この問題の求解に全数探索アプローチを適用することに多少希望があるように思われた。われわれは (heurstrategy=3 の設定により、積極的なヒューリスティクス戦略を採用した) Xpress の並列処理を実行させ続け、約 9 日間の計算 (経過) 時間で protfold の最適性を証明することができた。

この実証に使用したコンピュータは、3.0GHz Xeon プロセッサ 2 基、4GB RAM、Windows XP であった。Xpress の並列処理に使用したスレッド数は 4 だった (mipthreads = 4 の設定による)。このスレッド数にした特別な理由はないが、この問題についてのこれまでの経験から、ごく普通の選択である 2 スレッドにした場合よりも、4 スレッドの方が互いに協調して下界値の増加が速くなりそうであることは明らかであった。

Xpress により、計算時間 8 時間弱をかけ、探索木の 13 万 3,930 番目のノードで最適解が発見された。このノードでは、限界値はちょうど -35 であった。このギャップを完全に埋めるため、Xpress はさらに 240 万近くのノードを計算した。

問題 protfold の最適な目的値は -31 である。

5.4. rd-rplusc-21

問題 rd-rplusc-21 は、外部に反応器が付いた蒸留塔に相当する非線形モデルを線形化した緩和問題である。この問題は、変数の数に比べて多数の制約を有し、これらの制約はバイナリ変数の分枝によりアクティブとなったり、非アクティブとなったりする。この問題の求解のポイントは、変数の分枝法を選択するために良好な推定値を見つけることである。Xpress 2006B は、3 つの標準的な手順を使わないこと、すなわち、カットの生成 (cutstrategy=0)、ヒューリスティクス (heurstrategy = 0)、強力な分枝法 (sbiterlimit=0) の 3 つを作動させないことにより、この問題を解決できる。それに加えて、ソルバーは双対推定による精査に基づいた方式を使用して (sbestimate = 3)、最も有望な変数選択候補を選び出し、その選び出した候補の中から両方の分枝での擬似コストの低下量の和が最大である候補を選ばなければならない (varselection = 2)。こうした設定により、Xpress 2006B は (デュアル Xeon 3.0GHz を使って) 約 33 万 5,000 ノードを計算し、rd-rplusc-21 を 3.6 時間で解く。なお、rd-rplusc-21 モデルには GUB 制約に見られる 99 の主な決定変数が含まれ、99 の可能性すべてを列挙すると、作成された子問題はすべて 14 分で求解可能であることを最後に注記しておく。

問題 rd-rplusc-21 の最適な目的値は 165,395.2753 である (Vazacopoulos et al. 2006)。

5.5. sp97ar

問題sp97arは、van Hoeselら (2001) により提示された鉄道の路線計画モデルである。この問題の求解を試みると、全数探索のアプローチではうまくいかないことがすぐに明らかとなる。たとえ最良下界値を改善するような木探索の戦略が使われても、最良下界値の上昇ペースは、ほどなく極端に鈍化する。このLP緩和での下界値は6.5256e+8で、この値は徐々に6.578e+8にまで引き上げられるが、6.6e+8を上回らせることは不可能のように見える。

この問題の求解のポイントは、われわれが分枝限定法の変数固定と呼んでいる技法を使用することである。1万4,101のバイナリ変数のほとんどに対し、バイナリ変数の値を1に固定させることで、切除後のLPの下界値が顕著に引き上げられる。そのうえ、下界値が既知の最良解の値を上回ると、いくつかの変数は0に固定できる。したがって、各変数を順番に1に固定し、既知の最良解をカットオフ値に使用して限定された数のノードで木探索を実施することで、問題のサイズを縮小することができる。この木探索がノードの境界に到達せず、整数解を見つけることなく完了するならば、1にされていた変数は、0に固定された分枝限定であると言える。いくつかの変数は、数ノードが分枝限定の固定であればよい。しかし、この方法での変数の固定はますます困難となり、最終的には50万ノード近くの木探索が必要となる。しかしながら、1カ月を上回るCPU時間を使った後に変数の約90%はゼロに固定でき、問題は十分求解できるほど小さくなる。この最終段階の問題の求解でも、依然として24時間前後のCPU時間と強力な分枝法の利用が必要である。

問題sp97arの最適な目的値は、660,705,645.5である。

6. stp3dの初の実行可能解

MIPのstp3dは、MILP 2003の中で恐らく最も難しい問題である。この問題は、LP緩和と木ノードの再最適化の一部が困難であるようなスタイナー木の問題である。問題stp3dの初の実行可能解は、最大値低減戦略を用いた (つまり、varselection=4と設定することで) Xpressにより発見された。この解は、高速なノードの再最適化を適用するべく特別に設計されたXpress 2006Bを使い、約5時間で求められた。初めて知られることとなった問題stp3dの解は、目的値が529.778190である。

Xpressの次世代部分探索ヒューリスティクスを使用して、stp3dの最初の解はすぐに目的値が500.736である解へと改善された (§7を参照)。この解は3.3%の相対ギャップがあることに注意してもらいたい。

これまでのところ、stp3dの既知の最良解は500.736である。

7. すべてのオープンな問題のギャップの縮小

Xpress 2006Bに実装された、部分探索手順に基づいた次世代のXpressヒューリスティクスを使うことで、MILP 2003で最適解の定まっていなかった残る (7

表4. MILP 2003の未解決問題における目的値の改善

問題	Old best-known	Xpress improved	Gain
	旧来の最良の目的値 (z**)	Xpressが改善した目的値 (z*)	進歩 1-(z*)/(z**) (%)
stp3d	不明	500.736	n/a
ds	283.4425	116.59	58.9
momentum3	370,177.036	236,426.335	36.1
t1717	193,221	170,195	11.9
liu	1,172	1,102	6.0
dano3mip	691.2	687.733333	05

つの) 問題すべてにおいて、より最適に近い解が発見された。表4はこれらの改善を、われわれの知る限りの従前の既知の最良目的値とともに示している。

オープンな問題についての既知の最良目的値は、大半をMILP 2003ウェブサイトのディスカッションリストより採録した (Achterberg et al. 2006b)。ほかMILP 2003を参照している近年のいくつかの論文 (Achterberg and Berthold 2005, Balas and Saxena 2005, 2008, Fischetti and Lodi 2003, Hansen et al. 2006) も、同じく最良解を定めるための参照対象とした。

表4より、新たな部分探索ヒューリスティクスが、既知の最良解に対して相当程度改善を行えることが見てとれる。最も特筆すべき成果は問題dsで見られ、旧来の最良解よりも58.9%改善されている。

これらの解の1つ1つは、標準的なコンピュータで平均して数時間で求められた。厳密な求解時間のデータは手元にない。改善解は段階的な方法で求められたこと、また、2つの連続したヒューリスティクスに適用されるときのパラメータは通常それぞれ異なっていることがその理由である。

表5は、MILP 2003の残る未解決の問題を一覧にしたものである。各問題に対し、最適解への最良の上界値 (すなわち目的値) と下界値を示している。

momentum3, liu, dsについては、大きな相対ギャップがあり、それぞれ149.3%、79.8%、52.8%であることが見て取れる。これらの問題にはXpressにとって依然障壁があり、問題のロジックに関連した情報の活用や、MIPの技法に一層のブレイクスルーがなければ、求解の可能性は低いように思われる。

他方で、問題stp3dは比較的ギャップが小さい (3.3%)。だが分枝法や切除法によってギャップを縮小できるペースは比較的遅い。よって、この問題の解決にはしばらく時間がかかるであろう。

表5. MIPLIB 2003の未解決問題における既知の最良下界値および上界値

問題	下界値 (z^*)	上界値 (z^*)	ギャップ $ 1 - z^*/z^* $ (%)
stp3d	484.71817	500.736	3.3
dano3mip	578.05603	687.733333	190
t1717	136,538.4219	170,195	24.6
ds	76.32504272	1165	52.8
liu	613	1,102	79.8
momentum3	94,824.16406	236,426.335	1498

8. おわりに

この論考で、最も難易度の高い MIPLIB 2003 の問題のいくつかは、現行の技法を使用して求解可能であることを示してきた。特に、検討してきた求解手法、技法により、次の成果を得た。

(i) 5 つ の MIP 問題 (atlanta-ip、msc98-ip、protfold、rd-rplusc-21、sp97ar) は、初めて解が求められた。(ii) 1 つ の MIP 問題 (alc1s1) は、1 台のコンピュータで初めて求解できた。(iii) 問題 stp3d の初の実行可能解が求められた。(iv) MIPLIB 2003 の残るオープンな問題に対し、最適に近い解が求められた。

MIP の求解技法はこれまで大きく進歩を遂げてきたとはいえ、難易度の高い最適化問題の求解には、チューニング、モデリングが依然として重要な役割を果たしている。