Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

品野勇治 (YUJI SHINANO)

TOBIAS ACHTERBERG

TIMO BERTHOLD

STEFAN HEINZ

THORSTEN KOCH

STEFAN VIGERSKE

MICHAEL WINKLER

制約整数計画ソルバ **SCIP** の並列化
Parallelizing the Constraint Integer Programming Solver SCIP

Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

ZIB-Report 13-22 (April 2013)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

制約整数計画ソルバ SCIP の並列化*

Parallelizing the Constraint Integer Programming Solver SCIP

品野勇治 (Yuji Shinano)† Tobias Achterberg‡ Timo Berthold† Stefan Heinz†
Thorsten Koch† Stefan Vigerske§ Michael Winkler†

2013 年 4 月 22 日

概要

制約整数計画 (CIP: Constraint Integer Programming) は，制約プログラミング (CP: Constraint Programming)，混合整数計画 (MIP: Mixed Integer Programming), 充足可能性問題 (SAT: Satisfiability Problems) の研究分野におけるモデリング技術と解法を統合している．その結果，制約整数計画は，広いクラスの最適化問題を扱うことができる．SCIP (Solving Constraint Integer Programs) は，CIP を解くソルバとして実装され, Zuse Institute Berlin (ZIB) の研究者を中心として継続的に拡張が続けられている．本論文では，著者らによって開発された SCIP に対する 2 種類の並列化拡張を紹介する．一つは，複数計算ノード間で大規模に並列動作する ParaSCIP である．もう一つは，複数コアと共有メモリを持つ 1 台の計算機上で (スレッド) 並列で動作する FiberSCIP である．ParaSCIP は，HLRN II スーパーコンピュータ上で，一つのインスタンスを解くために最大 7,168 コアを利用した動作実績がある．また，統計数理研究所の Fujitsu PRIMERGY RX200S5 上でも，最大 512 コアを利用した動作実績がある．統計数理研究所の Fujitsu PRIMERGY RX200S5 上 では，これまでに最適解が得られていなかった MIPLIB2010 のインスタンスである `dg012142` に最適解を与えた．

Abstract

The paradigm of Constraint Integer Programming (CIP) combines modeling and solving techniques from the fields of Constraint Programming (CP), Mixed Integer Programming (MIP) and Satisfiability Problems (SAT). The paradigm allows us to address a wide range of optimization problems. SCIP is an implementation of the idea of CIP and is now continuously extended by a group of researchers centered at Zuse Institute Berlin (ZIB). This paper introduces two parallel extensions of SCIP. One is ParaSCIP, which is intended to run on a large scale distributed memory computing environment, and the other is FiberSCIP, intended to run on shared memory computing environments. ParaSCIP has successfully been run on the HLRN II supercomputer utilizing up to 7,168 cores to solve a single difficult MIP. It has also been tested on an ISM supercomputer (Fujitsu PRIMERGY RX200S5 using up to 512 cores). The previously unsolved instance `dg012142` from MIPLIB2010 was solved by using the ISM supercomputer.

1 はじめに

SCIP (Solving Constraint Integer Programs) は，制約整数計画 (CIP: Constraint Integer Program) を解くために開発されたソフトウェア・フレームワークである．CIP は，混合整数計画 (MIP: Mixed Integer Programming, 本論文では，混合整数線形計画: Mixed Integer Linear Programming を単に MIP と記述する) を完全に包含する最

* This paper will be published in Proceedings of the Institute of Statistical Mathematics (http://www.ism.ac.jp/editsec/toukei/index-e.html).
† Zuse Institute Berlin, Takustr. 7, D-14195 Berlin-Dahlem, Germany.
‡ ILOG, on IBM Deutschland GmbH, Ober-Eschbacher Str. 109, 61352 Bad Homburg v.d.H., Germany.
§ GAMS Software GmbH, P.O. Box 40 59, 50216 Frechen, Germany.

適化問題のクラスである (CIP の詳細は第 2 節で紹介する) . つまり，SCIP は MIP ソルバとしても機能し，H. D. Mittelmann の WEB ページ (http://plato.asu.edu/bench.html) に示されているベンチマーク結果を見ると，ソースコードが公開されている MIP ソルバの中では，現在，最高性能を示すソルバである．本節では，まず MIP ソルバ開発の現状を紹介する．なお，本論文では一般性を失うことなく，すべて最小化問題を対象として説明する．

ここでは，特定の問題専用ではなく，あらゆる MIP に対応できる汎用ソルバが対象である．過去 10 年間の MIP ソルバの性能向上は著しく，商用の MIP ソルバ，Gurobi(http://www.gurobi.com/), CPLEX(http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/), Xpress(http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx) が高性能であることは広く知られるようになってきている．Koch et al. (2011) には，過去 10 年間の MIP ソルバの性能向上を示す結果が定量的に示されている．MIP ソルバは基本的には分枝限定法の実装である．緩和問題として線形計画 (LP: Linear Programming) を解く分枝限定法は **LP ベースの分枝限定法**とよばれる．現在の MIP ソルバは，LP を解いた結果から得られる情報に基づく洗練されたアルゴリズムを実行する．

ソフトウェア的には，LP ベースの分枝限定法内で構成要素となる，前処理，分枝戦略などの機能は十分に整理されてきている．各機能の概略は第 2 節で紹介する．構成要素となる各機能において，どのようなタイミングで，どのアルゴリズムを，どの程度の頻度，どの程度の論理的な時間動作させるかは，大量の数値実験結果に基づいて決定されている．決定された内容は，実行時に自動調整されるようにプログラム中に組み込まれることもあるが，一般的にはパラメタ値として実行時に指定できるように実装される．その理由は，適切なパラメタ値は，問題のインスタンスに強く依存するためである．これまでに解けなかったインスタンスを解けるようにするため，それぞれの機能に新しいアルゴリズムが継続的に追加されている．新たに追加されたアルゴリズムが，あらゆるインスタンスに有効ということは稀であり，一般的には，そのアルゴリズムを動作させるべきかどうかを判定する手間を要することになり，これまでに解けていたインスタンスを解く時間はむしろ長くなる可能性もある．そこで，高性能な汎用ソルバを実現するためには，過去に高速に解けたインスタンスを含む，大量のインスタンスを用いた数値実験により，パラメタの再調整が必要となる．

一般的に，構成要素となる各機能に対して実装されるアルゴリズムのそれぞれがパラメタを持ち，SCIP を例にするとパラメタ数は 1000 を超える．この値は，商用ソルバよりも多いが，商用ソルバにも公開されていないパラメタが多数あるはずで，その実情は変わらないはずである (ユーザに対しては，全てのパラメタの設定には困難を伴うため，メタ・パラメタ設定が用意されるケースが多い．SCIP では，ヒューリスティックやカットなどに対して，「短時間の適用 (fast)」,「できる限り適用 (aggressive)」,「適用しない (none)」などを指定できる．). つまり，現在のソルバは，各機能毎に多くのパラメタを持ったヒューリスティックの塊が，大量のインスタンスによる数値実験結果に基づいて高度にチューニングされた大規模ソフトウェアである．実際，SCIP の現在のコード量は 450,000 行を超える．CPLEX は，コメント量は SCIP と比較して少ないにもかかわらず，コード量は 600,000 行を超える [Achterberg (2011)].

ソルバの性能を限られたインスタンス数で効率的に評価する上で，適当なインスタンスを収集するということは極めて重要である．ランダムに生成されたデータに基づくインスタンスは，極端に易しいか極端に困難なインスタンスとなることが多いため，現実問題，あるいは，具体的な組合せ最適化問題から生成されたインスタンスを商用ソルバ開発会社は持っており，データ自体が資産であるとも言える．研究目的としても，*MIPLIB*[Bixby et al. (1992) , Bixby (1998) , Achterberg et al. (2006) , Koch et al. (2011)] は，1991 年に公開されて以来，最新版の *MIPLIB2010* まで継続的に更新されてきた．このインスタンス・セットをまとめるという研究は，単なるデータの収集ではなく，ソルバ開発者の視点で何らかの意味を持つインスタンスが慎重に選択されライブラリに加えられてきている．その結果，極めて限られた数のインスタンス・セットにおいて，ソルバの性能が概観できるものとなっており，共通のインスタンス・セットによってソルバの性能が評価されてきたことが，現在の高性能ソルバ開発につながっている．

本論文で扱う SCIP の並列化は，基本的には分枝限定法における木探索の並列化である．しかし，SCIP により解か

れる問題のクラスは MIP を含んでおり，前述のような大規模ソフトウェアである MIP ソルバの性能を維持し，その各機能の性能を継承して動作しなければ，大規模な並列探索が実現されたとしても逐次処理ソルバの性能すら達成することは困難である．実際に，一般的な MIP ソルバの並列化に関する研究は過去にも行われてきている [Bixby et al. (1999) , Bussieck et al. (2009) , Chen and Ferris. (1999) , Eckstein (1997) , Phillips et al. (2006) , Linderoth (1998) , Ralphs et al. (2003) , Shinano et al. (2003, 2007, 2008) , Xu et al. (2009)] が，スーパーコンピュータを利用しても，商用の逐次ソルバの性能を超えることはほとんどなかった．著者らの知る限り，商用ソルバで解けないインスタンスを商用ソルバより先に解いたケースは，ほぼ手動で部分問題を GRID 上で解いた Bussieck et al. (2009) のみである．しかし，それは商用ソルバ CPLEX の探索木データを保存するファイルを解析して部分問題を生成し，分散計算させるもので，CPLEX が木構造データの出力を中止した時点で利用不可となった．SCIP に限らず，MIP ソルバの並列化の困難さは Ralphs (2006) に整理されている．

2005 年以降，単独 CPU コアの性能は大きく向上しなくなり，複数コアを持つ CPU がデスクトップ環境でも用いられるようになるにつれ，商用ソルバの並列化が活発になる．並列処理を適用できる環境がどこにでも存在するという環境の変化は大きく，商業的には環境の変化への追随が必然であったはずである．最も劇的に，絶妙のタイミングで開発されたのは Gurobi である．それまでは，単独の CPU コアの性能向上が直接ソルバの性能向上に寄与したが，複数コアを利用する場合には，プログラム構造の変更を伴う．各社，既存のコードを拡張して並列ソルバを実現していたが，Gurobi は設計の当初から並列実行を前提に設計された．その基本的なコンセプトは，“Tree of Trees Concept” と呼ばれ，探索木を部分木の集まりとみなし，部分木は既に高性能な MIP ソルバ自身に解かせるというコンセプトである．これは，著者らが試みてきた手法 [Shinano et al. (2008)] をソルバ内部に取り込んだコンセプトとみなすことができる．本論文で紹介する SCIP の並列化も同様の手法であり，並列探索木の様子は図 1 のようになる．図 1 において，影のついた各領域が部分木であり，部分木のルートとなる部分問題が独立して動作する MIP ソルバに転送され解かれる．

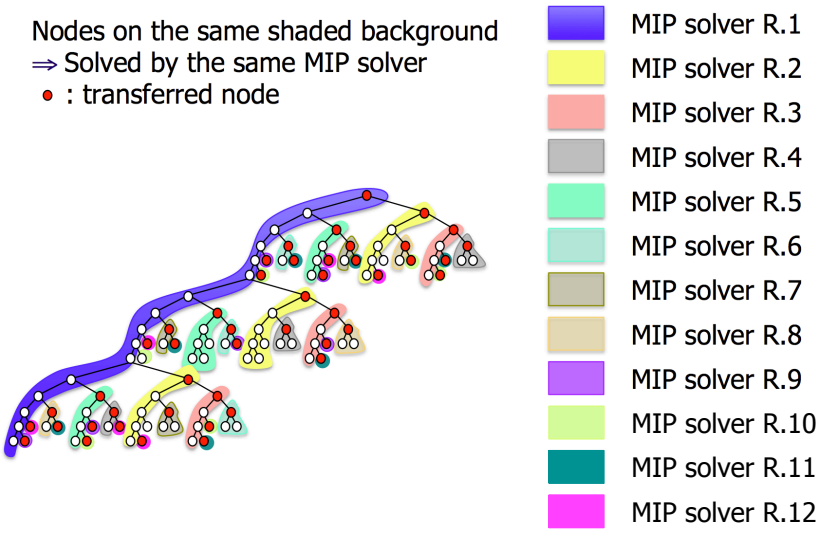

図 1　Parallel search tree generated by ParaSCIP and FiberSCIP

SCIP は，新しいアイデアをソルバ内部の機能毎に plug-in として柔軟に組み込める構造を持っている．実際，標準で提供されている MIP ソルバとしての機能ですら，plug-in として実現されているものが多く，構造的には簡単に置き換え可能な作りとなっている．しかし，内部に並列処理を組み込むことは当初の設計には含まれていなかった．スレッドセーフな設計にはなっていたが，実装はスレッドセーフでない部分も存在し，本研究を通して，そのような部分が検出され修正された．そのため，全ての商用ソルバが実現している共有メモリを利用した並列版さえ存在しな

かった．本論文では，著者らが開発した SCIP の並列化拡張を紹介する．この並列化により，SCIP は共有メモリを利用した並列ソルバだけでなく，大規模分散メモリ環境上で動作するソルバとなり，最適解が知られていなかった MIPLIB のインスタンスのいくつかに初めて最適解を与えることができた [Shinano et al. (2012)].

次節以降の構成は次の通りである．第 2 節では CIP と SCIP を紹介する．第 3 節では SCIP の並列化を実現しているソフトウェア・フレームワーク UG (Ubiquity Generator) を紹介し，UG 上に開発された ParaSCIP と FiberSCIP を紹介する．また，UG の機能として実現される並列処理について説明する．第 4 節では数値実験の結果を示す．数値実験は，MIP を対象とし，共有メモリ上での実行結果と，大規模分散メモリ環境上での実行結果を紹介する．第 5 節では今後の課題等について述べる．

2 制約整数計画 (CIP) と SCIP

制約整数計画は次のように定義される最適化問題である．

定義 1（制約整数計画 (CIP: Constraint Integer Program)）　制約整数計画 CIP $= (\mathfrak{C}, I, c)$ は，次の問題を対象とする:

$$\text{(CIP)} \quad c^\star = \min\{c^\top x \mid \mathfrak{C}(x),\ x \in \mathbb{R}^n,\ x_j \in \mathbb{Z} \text{ for all } j \in I\}.$$

ここで，$\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ は制約式 $\mathcal{C}_i : \mathbb{R}^n \to \{0,1\}$, $i = 1, \ldots, m$　からなる有限集合, $I \subseteq N = \{1, \ldots, n\}$ は変数の添字集合の部分集合, $c \in \mathbb{R}^n$ は目的関数の係数ベクトルである．加えて，CIP は次の条件を満たさなければならない:

$$\forall \hat{x}_I \in \mathbb{Z}^I\ \exists (A', b') : \{x_C \in \mathbb{R}^C \mid \mathfrak{C}(\hat{x}_I, x_C)\} = \{x_C \in \mathbb{R}^C \mid A' x_C \le b'\}. \tag{2.1}$$

ここで，$C := N \setminus I$, $A' \in \mathbb{R}^{k \times C}$, $b' \in \mathbb{R}^k$ であり，$k \in \mathbb{Z}_{\ge 0}$ である．

条件 (2.1) が要請しているのは，全ての整数変数の値が固定された後に残る部分問題は，常に LP になることである (注：SCIP が MINLP をサポートするように拡張された際に CIP の定義は拡張され，整数変数の値が固定された後に残る部分問題に対して最適解を求める手段があれば，LP である必要は無くなっている)．よって，整数変数だけが参照されるような制約式に対しては，二次式はもちろん，あらゆる非線形制約が許される．MIP は，全ての制約が線形，つまり，$\mathfrak{C}(x) = \{x \in \mathbb{R}^n \mid Ax \le b\}$，$A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ である，CIP の特殊ケースであり以下で定義される:

$$\text{(MIP)} \quad c^\star = \min\{c^\top x \mid x \in \mathbb{R}^n, x_j \in \mathbb{Z} \text{ for all } j \in I\}.$$

この MIP の最適値に対する下界値は，整数条件を取り除いた緩和問題である次の LP の最適値により与えられる:

$$\text{(LP)} \quad \overline{c^\star} = \min\{c^\top x \mid x \in \mathbb{R}^n, Ax \le b\}.$$

SCIP (Solving Constraint Integer Programs) は，CIP を解くために開発されたソフトウェア・フレームワークである．CIP は問題記述能力の極めて高い最適化問題であり，現在の SCIP では，CIP として記述できる全ての最適化問題が解けるわけではない．しかし，SCIP は必要になった新たな制約式のクラスを扱うことができるように，必要に応じて plug-in を追加することで拡張が可能なソフトウェア・フレームワークである．その中心的な働きをするのは，制約式ハンドラ (constraint handler) であり，制約式 $\mathfrak{C}(x)$ のクラスに対する plug-in を追加することで，扱える制約式のクラスが増える．この plug-in は，扱うクラスの制約式の意味を解釈し，そのクラスに属する制約式に対する操作を実装する．MIP の場合は，線形制約式ハンドラ (linear constraint handler) が線形制約式を扱う．制約式ハンドラの主な機能は，解が与えられた際に，解いているインスタンス中で，その制約式ハンドラが扱う制約式のクラスに属する全ての制約が満足されているかどうかをチェックすることである．与えられたインスタンスに含まれる全ての制約式のクラスに対する制約式ハンドラが用意されれば，全ての整数変数が固定されると LP になるため，この機能だけで列挙解法が構成できる．しかし，その解法は全列挙と同じであり，極めて遅い．そこで，SCIP の LP

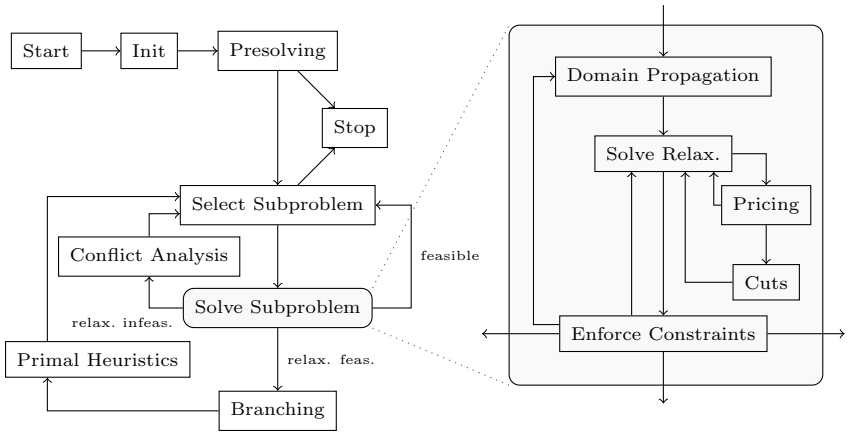

図 2　Flowchart of SCIP

ベースの分枝限定法のフレームワーク側から必要なタイミングで，コールバック・ルーチンが呼び出される．例えば，LP 緩和問題を解いた後，制約式ハンドラのコールバック関数が呼び出される．ここでは，緩和問題の解を調べ，各変数の定義域の縮小や，カットとなる制約式を生成し追加することなどが実現できる．

SCIP により実現される LP ベースの分枝限定法の主たる処理の流れを図 2 に示す．構成要素となる主たる機能は次の通りである．

- Presolvers: 問題のサイズを縮小する前処理
- Separators: 下界値を強化するカット生成
- Propagators: 変数の上界値・下界値の強化 (制約式ハンドラとは別途行われる部分)
- Branching Rules: 分枝変数の選択方法の指定
- Node Selectors: 分枝限定木における葉ノードの中で次に実行するものを指定
- Primal Heuristics: より良い上界値を与えるためのヒューリスティック解法
- Relaxation Handlers: 下界値を与える緩和問題の解法

これらの各機能単位に，plug-in として既に多くのアルゴリズムの実装が用意されており，独自に追加することも可能である．各構成要素となる機能は，図 2 に示される，SCIP が実行する LP ベースの分枝限定法の中の事前に決められた箇所において，事前に決められた順番で呼び出される．ユーザは自由に plug-in を追加できるが，デバッグモードにおいて，提供されている各関数は呼び出し可能なタイミングであるかどうかのチェックが行なわれるため，ユーザが事前に決められた順序を無視したプログラムを書くことは禁止される．一つの構成要素となる機能に対して複数の plug-in が存在し，その呼び出し順はパラメタで制御できる．

上記の機能以外にも，特に SCIP において特徴的な機能として，Conflict Analysis や Pricer がある．Conflict Analysis は，SAT ソルバで利用されてきた，探索過程で実行不可能となった部分問題から学習する仕組みを MIP へ拡張した機能であり，plug-in として拡張可能である．Pricer は，列生成法の実装を可能とするもので，列生成時の変数の追加を可能とする．CIP, SCIP に関するより詳細な内容に関しては，Achterberg (2007) を参照されたい．ここに述べていない plug-in の機能や plug-in の開発方法に関しては，SCIP の WEB ページ (http://scip.zib.de/) のドキュメントを参照されたい．

3 UG (Ubiquity Generator framework) を利用した並列ソルバ ParaSCIP と FiberSCIP

SCIP の並列化は，Tree of Trees Concept を特定のソルバに依存せず実現する単一のソフトウェア・フレームワーク *Ubiquity Generator(UG) framework* 上で実現している．本論文では，与えられた部分問題を MIP として解く，独立して動作するプログラム単位を単に SOLVER とよぶ．SOLVER は，一つのプロセッサ・コアに対して複数動作させることも可能であるが，通常は，プロセッサ・コアあたり，1 SOLVER を動作させる．実行時の SOLVER の動作形態は，通信がどのように実現されるかに応じて，スレッド，または，プロセスとして動作する．つまり，本論文で扱う並列ソルバの実体は，一つの問題を解くために，複数の SOLVER が同時に協調して動作するものである．

ここでは，まず，UG を紹介し，UG 上に開発された ParaSCIP と FiberSCIP を紹介する．UG は，C++ で記述されたソフトウェア・フレームワークである．並列化対象となる逐次，あるいは，共有メモリ上でスレッド並列に動作する分枝限定法による最適化ソルバ (SCIP に限らない) を，分散メモリ並列処理環境で動作させることを意図して開発された．基本的には，並列化対象ソルバ自身を多数並列に動作させ，それらの間の動的負荷分散を実現する．通信処理関係の関数に関しては，UG 内部で再定義されているので，通信を扱うクラスの継承クラスを開発することで，他の動作環境への移植を容易にしている．同様に，ソルバに関しても，特定のソルバに依存しない作りとなっている．主たる設計思想は，最新の高性能最適化ソルバの性能を維持した並列ソルバの開発を容易にすることである．

一般に，分枝限定法を実行する並列ソルバは，次の三つの実行フェーズを持つ [Ralphs (2006)].

1. *Ramp-Up* **フェーズ***:* 計算開始から，全ての SOLVER が同時に動作可能となる十分な部分問題群が生成されるまでの期間
2. *Primary* **フェーズ***:* 動作している SOLVER 内に十分な部分問題群が存在して，全ての SOLVER において並列探索が実現されている期間
3. *Ramp-Down* **フェーズ***:* 計算終了が近づいてきたため，全ての SOLVER が十分な部分問題を生成できず，常に，少なくとも一つの SOLVER は，処理する部分問題が得られない状態 (**遊休状態**) が生じている期間

UG の主たる機能は，動的負荷分散であるが，その他に，複数の Ramp-Up 手法を実現する機能，および，チェックポイント・再スタート機能を，フレームワークの機能として提供する．前述のように，通信手段は選択可能な構造となっているので，現在は，共有メモリ上でのマルチ・スレッド並列による比較的小規模の並列処理を実現するために Pthreads ライブラリ，分散メモリ環境上で大規模並列処理を実現するために MPI (Message Passing Interface) ライブラリが SCIP ソルバの並列動作に利用可能である．UG を利用して開発されたソルバの命名規則は次の通りである:

ug [並列化対象ソルバ名, 通信に利用するライブラリ名].

この命名規則は並列ソルバの特徴を直接的に表現している．この命名規則を使うと，ParaSCIP は ug [SCIP,MPI] であり，FiberSCIP は ug [SCIP,Pthreads] である．UG を利用することにより，論理的には全く同じ並列化手法を，異なる動作環境で実行することができる．このような実装により，論理的な挙動と動作環境の影響を，より明確に詳細に調べる手段が提供できる．

3.1 UG が提供する並列処理

ここでは，UG が提供する機能により，どのような並列処理が実現されるのかについて，計算の開始から終了までの流れの中で説明する．ここで説明する並列処理は，プログラムの実行時形態としてプロセスとスレッドの違いはあるが，論理的には，ParaSCIP と FiberSCIP の両方で共通である．プログラム実行時には，ParaSCIP の場合 2 種類

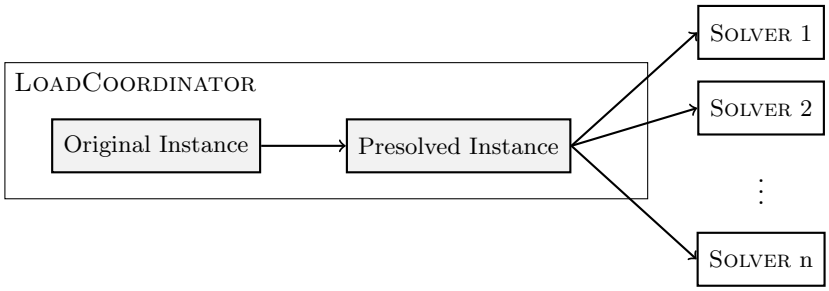

図 3　Initialization step

の MPI プロセス，FiberSCIP の場合 2 種類のスレッドがシステム中に存在する．一つは，既に述べた部分問題を解く Solver であり，もう一つは，Solver 間の負荷分散を司り，プログラム全体を制御する LoadCoordinator である．

まず，初期化処理について説明する．プログラムが起動すると，LoadCoordinator が問題のデータを読み込む．読み込んだ問題データは，LoadCoordinator 内で，一度，前処理 (presolve) される．現在の MIP ソルバの前処理は極めて強力で，固定可能な変数の値の固定，冗長な制約式の除去，変数の上・下界値の強化等を行い，問題のサイズが劇的に小さくなることが多い．本論文では，前処理前の問題を**元インスタンス** *(Original instance)*，前処理後の問題を **前処理後インスタンス** *(Presolved instance)* とよぶ．図 3 に示すように，LoadCoordinator は，前処理後インスタンスを全 Solver へ転送する．各 Solver は，前処理後インスタンスを Solver 内部では元インスタンスとして扱う．つまり，部分問題は，Solver 内部で，再度，前処理される．この前処理後インスタンスの転送が，プログラム実行中で行われる最も大量のデータ転送になるが，それは Solver が初期化される，この 1 回のみである．各 Solver は，前処理後インスタンスを Solver 内部に保持し，部分問題の表現として基本的には前処理後インスタンスと部分問題における変数の上・下界値変化の差分だけを保持する．つまり，部分問題は，計算過程では常に前処理後インスタンスを対象として解かれ，最終的に求まった解は，LoadCoordinator 内に保存されている前処理時の情報を利用して，元インスタンスに対する解へ変換され出力される．

初期化が終わると，LoadCoordinator は分枝限定木のルートとなる部分問題を作成する．このルートとなる部分問題は前処理後インスタンスと同じなので，UG が利用する管理情報を除く部分問題としての情報 (前処理後インスタンスとの差分) は空である．この部分問題の情報を含む転送されるデータを ParaNode とよぶ．まず，ルートとなる ParaNode が Solver に転送され計算を開始する．図 1 において，ルートノードが転送されたノードとなっている点に注意されたい．LoadCoordinator 内部では，転送された部分問題が解き終わるまで ParaNode は Solver の状態を管理するデータ構造内に保管される．各 Solver は，ParaNode を受け取ると，Solver 内部に保持している 前処理後インスタンスと ParaNode の情報により，完全な部分問題を生成して，それを解き始める．

UG は，2 種類の Ramp-Up 手法を提供する．

Normal Ramp-Up　ParaNode を受け取って，遊休状態から部分問題を解いている状態 (**活性化状態**とよぶ) となった Solver は，分枝毎に，生成される二つの部分問題の一方を ParaNode へ変換して LoadCoordinator へ転送し，もう一方は，Solver 内部で継続して解く．LoadCoordinator は，その内部に**ノード・プール**を持ち，ParaNode を受け取ると，未処理の ParaNode をその下界値の昇順に保持する．ただし，遊休状態の Solver が存在する限り，ただちに，ParaNode は遊休状態の Solver に割り当てられる．また，遊休状態の Solver が存在しない場合でも，ノード・プールに保存されている「**良好な**」ParaNode の数が，実行時パラメタで指定された値 p を下回る間は， ParaNode の収集を継続する．ここで，「良好な」ParaNode とは，その部分問題の下界値と，LoadCoordinator が管理する全ての Solver の計算過程で得た下界値の最小値 (これを GlobalDualBound とよぶ) との差が，実行時パラメタで指定された相対誤差範囲内にある

ものを指す．良好な ParaNode の数が p を超えると，全ての Solver に Ramp-Up 終了の通知を転送する．各 Solver は，その通知を受け取ると，ParaNode の作成と転送を中止する．

Racing Ramp-Up LoadCoordinator は，ルートとなる ParaNode を全ての Solver に転送する．それと同時に，Solver 数の異なるパラメタセットを生成し，それぞれを各 Solver に転送する．そして，各 Solver は受け取ったパラメタセットにより，独立にルートとなる問題を解き，異なる分枝限定木を生成する．各 Solver における下界値や生成されている部分問題の数，あるいは，制限時間などの情報 (実行時パラメタで指定される) により，勝者となる Solver が LoadCoordinator により決定される．勝者が決まれば，LoadCoordinator は当該 Solver へ勝者となったことを伝える．勝者となった Solver は，生成されている未処理の部分問題を ParaNode に変換して，LoadCoordinator からの送信停止指示を受け取るまで転送する．一方，LoadCoordinator は，勝者以外の Solver に対しては，計算の停止を要求し，その要求を受け取った Solver は実行中の計算を破棄し，新たな部分問題が転送されるのを待つ．本論文では，全ての Solver が独立にルート問題を解いている段階を，*Racing* **段階**とよぶ．通常，動作している Solver の全てに部分問題を与えることが可能になるまで，Racing 段階が継続するようにパラメタを設定して動作させるが，実行環境と扱うインスタンスによっては，そのような設定が現実的でない場合がある．例えば，7,000 Solver を動作させているが，勝者が 7,000 の未処理の部分問題を生成するのを持つのに数日を要する場合もある．そのような場合には，Racing 段階は，十分な部分問題が無い状態で終了し，その後の処理は，Normal Ramp-Up と同じになる．Racing Ramp-Up は，大規模な並列処理を適用した際に，遊休状態にある Solver を利用して，パラメタの学習，あるいは，チューニングを動的に行うことを意図して設計された．

動的負荷分散方式は，基本的には Shinano et al. (2008) で設計された方式を踏襲している．MIP を解く上での負荷分散は，最適値に対する上界値と下界値に強く依存する．未処理の部分問題数だけによる単なる負荷分散では，本来解く必要の無い部分問題の計算に時間を費やすことになる．上界値は，暫定解 (それまでに見つかっている実行可能解) よりも小さな目的関数値を持つ実行可能解が見つかることで更新される．ある Solver において，暫定解が更新されると，その解は LoadCoordinator に転送され，その目的関数値，つまり，上界値は，できる限り迅速に全ての Solver へ伝達され，各 Solver では受け取った上界値により計算不要となる部分問題が削除される．

各 Solver は，定期的に，その状態を LoadCoordinator へ転送する．Solver の状態を示す情報は，その Solver 内の下界値，解いた部分問題数，未処理の部分問題数などであり，このデータ量は極めて少ない．伝達の頻度は，実行時パラメタで指定する．この情報により，LoadCoordinator は全ての Solver の状態を把握する．Racing 段階においても，この情報は通知され，この情報に基づき勝者の決定が行われる ．下界値は，分枝限定木の各ノードにおいて，LP を繰り返し解いた結果から求められる．各 Solver は，この情報を転送すると同時に，LoadCoordinator から，LoadCoordinator 中の ノード・プールに存在する ParaNode が持つ下界値の最小値 (これを BestDualBound とよぶ) を受け取る．BestDualBound には，Solver で解いている部分問題に関する情報は含まれていない．

LoadCoordinator がノード・プールに ParaNode を持つ限り，遊休状態の Solver が生じると ParaNode が即座に転送される．よって，同時に複数の Solver が遊休状態となった場合にも，常に Solver へ部分問題を割り当てるためには，LoadCoordinator は，常にある程度の ParaNode を保持する必要がある．そこで，少なくとも p 個の「良好な」ParaNode を LoadCoordinator 中のノード・プールに保持するため，Shinano et al. (2008) と同様に，**収集モード** *(collecting mode)* を導入した．ここでも，「良好な」とは，その部分木における下界値 (NodeDualBound) と，全ての Solver で生成されている探索木を考慮した下界値 (GlobalDualBound) の差が小さいという意味である．

もし，LoadCoordinator が収集モードではなく，次のような条件:

$$\frac{\text{NodeDualBound} - \text{GlobalDualBound}}{\max\{|\text{GlobalDualBound}|, 1.0\}} < \text{Threshold} \tag{3.1}$$

を満たす「良好な」ParaNode を p 個未満しか ノード・プールに持たない場合，LoadCoordinator は収集モードとなり，条件 (3.1) を満たす部分問題を持つ Solver へ 収集モードになるよう要求を出す．この要求を受け取った Solver は収集モードとなる．現在のソルバは 一般的に探索規則を柔軟に変更できるので，収集モードとなったソルバは，その Solver 内で実行中の分枝限定木の探索規則を保存し，探索規則を下界値優先探索へ切り替え，LoadCoordinator から収集モードを停止するように要求が来るまで，分枝毎に 1 個の部分問題を ParaNode へ変換して LoadCoordinator へ転送する．LoadCoordinator は，「良好な」ParaNode の数が $m_p \cdot p(m_p$ も実行時パラメタ) 以上になると，収集モード停止要求を収集モード中の Solver へ送る．Solver は，収集モード停止要求を受け取ると，転送を中止し元の探索規則に切り替える．

分枝限定法では，良い上界値が求まれば限定されるはずの部分問題が限定されずに計算されることがある．このような不要な計算は，大規模な並列処理を実現した場合に助長されることが多い．理由の一つは上界値の伝達の遅れであり，もう一つは，ある Solver が極めて悪い下界値の部分問題しか持たないという状態が容易に生じるためである．UG では，この悪い下界値の部分問題しか持たない状態は，LoadCoordinator のノード・プールにある部分問題の下界値 BestDualBound と，Solver 内で計算中の部分問題の下界値を比較することで判断できる．BestDualBound の値が，Solver で計算中の部分問題の下界値に比べて極めて小さい場合には，LoadCoordinator 中の ParaNode の計算へ移行した方が良い．そこで，Solver は，新たな ParaNode を LoadCoordinator へ要求し，新たな ParaNode を受け取ると，計算中の分枝限定木を破棄し，受け取った ParaNode の計算を開始する．LoadCoordinator は，複数の Solver から要求を受ける可能性があるので，常に ParaNode 要求のあった Solver へ ParaNode を送れるわけではない．新たな ParaNode を受け取れなかった Solver は，そのまま計算を続行する．一方，LoadCoordinator が新たな ParaNode を送る場合には，Solver で解かれていた ParaNode は，ノード・プールへ戻され，Solver で計算されていた部分問題の計算は遅らされる．

計算終了フェーズは，LoadCoordinator が，全ての Solver が遊休状態であり，ノード・プールに ParaNode が無いという状態を検出すると始まり，全ての Solver に統計情報の転送を要求する．それらが集まると，解と統計情報を出力して計算が終了する．

3.2 チェックポイントと再スタート

分枝限定法は，一般的に計算終了時間の予測が困難である．スーパーコンピュータでは，各ジョブに実行時間の上限が設定されているのが一般的である．そのため，何らかのチェックポイントと再スタートの機能を利用することが必然となる．本論文が扱う問題は，解くことが困難な問題であり，巨大な分枝限定木が生成されることを仮定している．それらを定期的にチェックポイントファイルへ全て書き出すと，時間を要する上に計算機システムに対しても大きな負荷を与えることになる．そこで，UG は専用のチェックポイントと再スタート機構を持つ．

UG の LoadCoordinator は，分枝限定木全体は保持せず，Solver へ転送された部分問題だけを保持し，その数は，未処理の部分問題数全体と比較すると極めて少ない．そこで，LoadCoordinator 中の部分問題だけをチェックポイントで保存することが考えられる．しかし，ある Solver で生成された部分問題が他の Solver へ転送されるため，これらの部分問題間に祖先・子孫関係が存在する場合がある．再スタート時に，それらを読み込むと，祖先とその子孫となる部分問題がチェックポイントファイル中に存在する場合，子孫から生成される部分木が冗長に解かれることとなる．従って，LoadCoordinator 中では，部分問題間の祖先・子孫の関係を常に追跡し，祖先が LoadCoordinator 中に存在しない部分問題だけをチェックポイント時に保存する．この部分問題の情報は，具体的には ParaNode の情報であり，変数の上界値・下界値の変更だけを保持しているのでデータ量は極めて少ない．その他には，解法が動作している状態における統計情報である．この統計情報は，各 Solver が送受信した部分問題数，各 Solver で生成された分枝木のノード数などである．インスタンスデータそのものは，チェックポイント時に

は保存されない．

再スタート時には，インスタンスデータは，再度読み込まれ，その後，チェックポイントファイルから部分問題群が読み込まれ計算が再スタートする．このような再スタートは，大量の計算結果を破棄し，再スタート時にはチェックポイントの状態を回復するために多大な時間を要することになる．このような自明な欠点はあるが，一方でチェックポイントファイルに保存された部分問題の下界値は，その部分問題が生成された時点よりも正確な値に更新されている．再スタート時には，その更新された情報により部分問題は並べ替えられ，その順に SOLVER へ送られて解かれるため，チェックポイント時点と全く同じ状態は生成せず，より最適解が得られる可能性の高い方向への探索が進むという解法上の利点がある．定量的な評価は極めて困難であるが，再スタートを繰り返すことで，商用ソルバでは発見することが困難な実行可能解がこれまでに何度も見つかっているため (具体的には，2009 年当時オープンであった MIPLIB2003 のインスタンス 6 問中，5 問において最良解を更新している．1 問は既知の最良解が最適解であった)，真に最適解を求めることが困難な問題ではかなり有効に機能しているのではないかと思われる．

3.3 ParaSCIP と FiberSCIP の違いについて

並列化は UG フレームワークによって実現されるため，ParaSCIP と FiberSCIP の最も大きな違いは，SOLVER の実行形態だけであり，論理的には全ての機能が共通となる．しかし，実装上は全てのデータが共有メモリ内にある FiberSCIP と，共有メモリを持たない環境での動作を実現している ParaSCIP では，SCIP の拡張に対する柔軟性に対して大きな差が生じる．

FiberSCIP では，共有メモリに存在する SCIP の環境に常にアクセスでき，SCIP の API (Application Program Interface) を直接利用して，並列ソルバを実現している．SCIP の機能が拡張した場合には，その拡張に関連する API は必然的に実装されるので，SCIP の拡張と同時に FiberSCIP の機能も拡張する．一方，ParaSCIP の場合は，SCIP の環境から必要なデータを取り出し，UG フレームワーク内で利用するデータへ変換後，転送し，転送先では逆に UG フレームワーク内のデータから SCIP の環境への設定が必要となる．SCIP の拡張に対する柔軟性を最も損ねるのは，前処理後インスタンスデータを転送する部分である．SCIP は，制約式ハンドラ が扱うデータ形式を決め，そのクラスに属する制約式のデータを管理している．新たな制約式ハンドラが追加され，サポートされる最適化問題のクラスが拡張された場合，現在の仕組みを維持して ParaSCIP が拡張されたクラスの最適化問題を解けるようにするには，新たなプログラムコードの追加を余儀なくされる．一方，FiberSCIP は，共有メモリ内に存在する SCIP の環境へのポインタのみを SOLVER 側へ渡し，SCIP が提供する環境をコピーする関数群を利用して，前処理後インスタンスの転送を実現している．従って，SCIP がサポートする最適化問題のクラスを拡張した場合には，自動的に FiberSCIP がサポートする最適化問題のクラスも拡張する．

SOLVER 間で転送するデータの種類を追加し，SOLVER 間で共有する情報により並列ソルバの性能向上を図る場合，プログラム開発とデバッグは，通常のデバッガ利用できるので FiberSCIP の方が格段に容易である．そして，デバッグを終えた後は，わずかなコードの追加により大規模分散メモリ並列環境で ParaSCIP として動作させることが可能である．FiberSCIP は，ソルバとしての性能だけが重要なのではなく，大規模分散メモリ並列環境上で動作させるための ParaSCIP の性能改善を容易に行える開発環境を与えている，という点が重要なのである．

4 数値実験結果

まず，MIPLIB2010 の Benchmark テストセット (http://miplib.zib.de/miplib2010-benchmark.php) を利用した FiberSCIP による数値実験の結果を示す．その後，ParaSCIP による大規模計算の代表的な結果をいくつか紹介する．

4.1 FiberSCIP の数値実験結果

数値実験に利用した MIPLIB2010 の Benchmark テストセット (87 インスタンス) の特徴を表 1 に示す．Benchmark テストセットは，MIP ソルバの性能を 100 未満のインスタンス数で概観することを意図して慎重に選ばれている．表 1 において，Vars. は変数の数，Cons. は制約式の数，NZs は制約行列の非ゼロ要素の数，Bin は 0-1 変数の数，Int は整数変数の数，Cont は連続変数の数である．表 1 には，元インスタンスと 1 回の前処理後インスタンスの情報が示されている．現在の MIP ソルバの前処理は問題の規模をかなり縮小することがわかる．最後のカラムの Vars.*Cons. は定常的に必要とするメモリ容量の見積もり値として利用するために示した．

表 1　Benchmark set for MIP (MIPLIB2010 Benchmark set)

	Original instance						Presolved instance						
Name	Vars.	Cons.	NZs	Bin	Int	Cont	Vars.	Cons.	Bin	Int	Impl	Cont	Vars.*Cons.
30n20b8	18380	576	109706	11036	7344	0	8028	403	7971	2	55	0	3235284
acc-tight5	1339	3052	16134	1339	0	0	998	2291	998	0	0	0	2286418
aflow40b	2728	1442	6783	1364	0	1364	1878	1144	940	0	0	938	2148432
air04	8904	823	72965	8904	0	0	7412	605	7412	0	0	0	4484260
app1-2	26871	53467	199175	13300	0	13571	26265	52555	13000	0	0	13265	1380357075
ash608gpia-3col	3651	24748	74244	3651	0	0	3651	24748	3651	0	0	0	90354948
bab5	21600	4964	155520	21600	0	0	21366	4819	21366	0	0	0	102962754
beasleyC3	2500	1750	5000	1250	0	1250	1704	1153	852	0	0	852	1964712
biella1	7328	1203	71489	6110	0	1218	7311	1202	6110	0	1197	4	8787822
bienst2	505	576	2184	35	0	470	449	520	35	0	0	414	233480
binkar10_1	2298	1026	4496	170	0	2128	1443	825	170	0	0	1273	1190475
bley_xl1	5831	175620	869391	5831	0	0	829	7988	829	0	0	0	6622052
bnatt350	3150	4923	19061	3150	0	0	1757	6587	1757	0	0	0	11573359
core2536-691	15293	2539	177739	15284	0	9	15269	1920	15269	0	0	0	29316480
cov1075	120	637	14280	120	0	0	120	637	120	0	0	0	76440
csched010	1758	351	6376	1457	0	301	1654	295	1457	0	1	196	487930
danoint	521	664	3232	56	0	465	513	656	56	0	0	457	336528
dfn-gwin-UUM	938	158	2632		90	848	936	156	0	90	0	846	146016
eil33-2	4516	32	44243	4516	0	0	4516	32	4516	0	0	0	144512
eilB101	2818	100	24120	2818	0	0	2818	100	2818	0	0	0	281800
enlight13	338	169	962	169	169	0	338	169	173	165	0	0	57122
enlight14	392	196	1120	196	196	0	392	196	200	192	0	0	76832
ex9	10404	40962	517112	10404	0	0	12	18	12	0	0	0	216
glass4	322	396	1815	302	0	20	317	392	298	0	0	19	124264
gmu-35-40	1205	424	4843	1200	0	5	560	325	555	0	0	5	182000
iis-100-0-cov	100	3831	22986	100	0	0	100	3831	100	0	0	0	383100
iis-bupa-cov	345	4803	38392	345	0	0	341	4803	341	0	0	0	1637823
iis-pima-cov	768	7201	71941	768	0	0	736	7201	736	0	0	0	5299936
lectsched-4-obj	7901	14163	82428	7665	236	0	2606	4801	2500	106	0	0	12511406
m100n500k4r1	500	100	2000	500	0	0	500	100	500	0	0	0	50000
macrophage	2260	3164	9492	2260	0	0	2260	3164	2260	0	0	0	7150640
map18	164547	328818	549920	146	0	164401	15398	31183	118	0	0	15280	480155834
map20	164547	328818	549920	146	0	164401	15398	31183	118	0	0	15280	480155834
mcsched	1747	2107	8088	1731	14	2	1495	1853	1495	0	0	0	2770235
mik-250-1-100-1	251	151	5351	100	150	1	201	159	107	93	1	0	31959
mine-166-5	830	8429	19412	830	0	0	746	7135	746	0	0	0	5322710
mine-90-10	900	6270	15407	900	0	0	800	4656	800	0	0	0	3724800
msc98-ip	21143	15850	92918	20237	53	853	12737	14986	11927	0	0	810	190876682
mspp16	29280	561657	27678735	29280	0	0	12737	14986	11927	0	0	810	190876682
mzzv11	10240	9499	134603	9989	251	0	6892	6816	6657	184	51	0	46975872
n3div36	22120	4484	340740	22120	0	0	19764	4454	19764	0	0	0	88028856
n3seq24	119856	6044	3232340	119856	0	0	119856	5950	119856	0	0	0	713143200
n4-3	3596	1236	14036		174	3422	3360	996	0	174	0	3186	3346560
neos-1109824	1520	28979	89528	1520	0	0	1520	9979	1520	0	0	0	15168080
neos-1337307	2840	5687	30799	2840	0	0	2840	5687	2840	0	0	0	16151080
neos-1396125	1161	1494	5511	129	0	1032	1158	1491	129	0	3	1026	1726578
neos13	1827	20852	253842	1815	0	12	1827	17320	1815	0	0	12	31643640
neos-1601936	4446	3131	72500	3906	0	540	3920	3105	3570	0	350	0	12171600
neos18	3312	11402	24614	3312	0	0	761	3300	761	0	0	0	2511300
neos-476283	11915	10015	3945693	5588	0	6327	11825	9583	5544	0	0	6281	113318975
neos-686190	3660	3664	18085	3600	60	0	3660	3658	3600	60	0	0	13388280
neos-849702	1737	1041	19308	1737	0	0	1713	996	1713	0	0	0	1706148
neos-916792	1474	1909	134442	717	0	757	1361	1408	708	0	0	653	1916288
neos-934278	23123	11495	125577	19955	0	3168	8121	8123	7354	0	767	0	65966883
net12	14115	14021	80384	1603	0	12512	12546	12787	1120	0	0	11426	160425702
netdiversion	129180	119589	615282	129180	0	0	128968	99483	128968	0	0	0	12830123544
newdano	505	576	2184	56	0	449	449	520	56	0	0	393	233480
noswot	128	182	735	75	25	28	120	171	75	20	0	25	20520
ns1208400	2883	4289	81746	2880	0	3	2596	1981	2596	0	0	0	5142676
ns1688347	2685	4191	66908	2685	0	0	1360	2599	1360	0	0	0	3534640
ns1758913	17956	624166	1283444	17822	0	134	17624	588479	17622	0	0	2	10371353896
ns1766074	100	182	666		90	10	100	110	90	0	0	10	11000
ns1830653	1629	2932	100933	1458	0	171	648	1489	514	0	134	0	964872
opm2-z7-s2	2023	31798	79762	2023	0	0	1900	26839	1900	0	0	0	50994100
pg5_34	2600	225	7700	100	0	2500	2600	225	100	0	0	2500	585000
pigeon-10	490	931	8150	400	0	90	400	525	370	0	0	30	210000
pw-myciel4	1059	8164	17779	1058	1	0	1036	4180	1035	1	0	0	4330480
qiu	840	1192	3432	48	0	792	840	1192	48	0	0	792	1001280
rail507	63019	509	468878	63009	0	10	62997	473	62997	0	0	0	29797581
ran16x16	512	288	1024	256	0	256	512	288	256	0	0	256	147456
reblock67	670	2523	7495	670	0	0	606	2130	606	0	0	0	1290780
rmatr100-p10	7359	7260	21877	100	0	7259	7359	7260	100	0	0	7259	53426340
rmatr100-p5	8784	8685	26152	100	0	8684	8784	8685	100	0	0	8684	76289040
rmine6	1096	7078	18084	1096	0	0	1084	7066	1084	0	0	0	7659544
rocII-4-11	9234	21738	243106	9086	0	148	1403	3734	1321	0	0	82	5238802
rococoC10-001000	3117	1293	11751	2993	124	0	2442	576	2442	0	0	0	1406592
roll3000	1166	2295	29386	246	492	428	862	1439	624	114	11	113	1240418
satellites1-25	9013	5996	59023	8509	0	504	7081	4203	7049	0	0	32	29761443
sp98ic	10894	825	316317	10894	0	0	10894	797	10894	0	0	0	8682518
sp98ir	1680	1531	71704	871	809	0	1557	1372	869	688	0	0	2136204
tanglegram1	34759	68342	205026	34759	0	0	34759	68342	34759	0	0	0	2375499578
tanglegram2	4714	8980	26940	4714	0	0	4714	8980	4714	0	0	0	42331720
timtab1	397	171	829	64	107	226	201	166	54	92	0	55	33366
triptim1	30055	15706	515436	20451	9597	7	21938	15679	12446	9492	0	0	343965902
unitcal_7	25755	48939	127595	2856	0	22899	20292	38651	2503	0	0	17789	784306092
vpphard	51471	47280	372305	51471	0	0	28966	24896	28966	0	0	0	721137536
zib54-UUE	5150	1809	15288	81	0	5069	5069	1761	80	0	0	4989	8926509

数値実験には，ZIB の Alibaba クラスタの同じタイプの 40 の計算ノードを利用した．各ノードは，2.5GHz の Quad-Core Xeon E5420 CPU を 2 個搭載する PowerEdgeTM 2950 でメモリ容量は 16GB である．数値実験に利用した SCIP のバージョンは 2.1.1 であり，LP ソルバとして SoPlex 1.6.0.1 を利用した．FiberSCIP の全ての数値実験は，2 時間の計算時間制限を設けて実施した．Racing Ramp-Up における Racing 段階の停止条件は，2 時間の制限時間の 0.5% にあたる 36 秒とした．ただし，300 個の未処理の部分問題を勝者 SOLVER が保持しない場合は，制限時

間の 25% にあたる最大 1800 秒まで Racing 段階を継続する．加えて，実行可能解が見つかっていない状態の場合，Racing 段階は制限時間まで継続する．これは，FiberSCIP では実行不可能性の検出に有効な情報を，他の Solver へ転送していないので，転送した部分問題が巨大な分枝限定木を生成する可能性が高いためである．本数値実験では，制限時間を考慮してこのような設定を行った．SCIP のパラメタ設定は，基本的にはデフォルトである．ただし，利用メモリ量は 1 Solver スレッド当たり 2GB で収まるように設定しており，数値実験を通して，ディスクアクセスを伴うページフォルトは生じなかった．

まず，逐次処理の SCIP との性能の違い，および，マルチスレッドで Solver を動作させた際の挙動を明確にするための予備実験を実施し，並列化の効果を測定するための基準となる逐次計算結果を準備する．表 2 は，逐次処理の SCIP と，Solver を 1 スレッドだけ動作させた FiberSCIP による実験結果である．LoadCoordinator スレッドは存在し，Solver の状態を伝える通信は行うが，Solver スレッドは一つなので逐次処理である．LoadCoordinator による前処理を禁止し，元インスタンスをそのまま Solver スレッドへ転送した場合 (Presolve once) と，通常の動作である，LoadCoordinator 側と Solver 側で計 2 回の前処理が実行された場合の結果が示されている．各設定における実験結果として，解かれた部分問題数と計算時間が示されている．計算時間に “>” が示されているのは制限時間を超えたことを示す．また，“!” が付加されている結果は，プログラムが異常終了したことを示す (制限時間を超えた場合と異常終了の場合は，解かれた部分問題数が正しく出力できないことがあり，正しく出力できない場合にも 1 と表示されている) ．最後の Solution Status は，SCIP および二つの FiberSCIP での計算結果の最悪の計算の終了状態と，正常終了した場合には解の状態を示す．この解の状態は，実行不可能問題の場合には実行不可能であることを出力したら “ok”．最適解をもつ問題の場合に，計算が中断したとき，実行可能解をみつけていれば “ok”，計算が終了し最適値が必要な精度で一致していれば “ok” を示している．MIPLIB2010 には，解のチェッカーが付属しており，生成された解の実行可能性等がチェックできる．本論文に示されている FiberSCIP の全ての数値実験結果は，何らかの解チェッカーにより解が必要な精度で求められているかが常にチェックされている．表の下部には，“解けたインスタンス数 (solved)/制限時間を超えたインスタンス数 (timelimit)/異常終了したインスタンス数 (abort)” が示されている．また，各カラムに対して，幾何平均の値が示されている．この幾何平均の計算には，計算対象のうち 3 ケース全てにおいて解けたインスタンスの結果だけを用い，その解けたインスタンス数は幾何平均の行の最左端に示されている．解けたインスタンス数等の情報の次の行は，3 ケース全てを計算対象としており，全てのケースで解けたインスタンス数は 56 である．それに続く 2 行に示されているのは，左側から 2 つのケースを計算対象とした場合と，右側から 2 つのケースを計算対象とした場合である．最下行の幾何平均は，時間制限に達した，または，異常終了した場合を全て制限時間 (7200 秒) として計算した計算時間の幾何平均であり，H. D. Mittlemann によるベンチマークで採用されている評価方法である．本論文における FiberSCIP による数値実験結果の表は，以降同じ表記方法はであり差分を適宜補足する．

SCIP と前処理を 1 回だけ行った FiberSCIP の結果は極めて近かった．前処理を 1 回だけ行う場合には，SCIP の環境へ読み込まれたデータが，単に Solver 側の環境にコピーされて計算が始まる．計算時間は LoadCoordinator との通信のオーバーヘッドを含むため動作環境の状態によって若干異なるが，ほとんどの結果においてほぼ同じ値を示し，通信のオーバーヘッドが無視できることがわかる．一方，解かれる部分問題数に関しては SCIP と同じになることが自然であるが，実際には若干の違いを生じた．これは，SCIP のパラメタによりメモリ使用量を制限しているため，探索が深さ優先に切り替わるタイミングの違いから生じていると考えられる．本来の FiberSCIP の実装である 2 回の前処理が実行される場合には，結果は SCIP と比較して大きく異なった．表 2 の下部に示されるように，幾何平均で 47 秒程度遅くなっている．各インスタンスによる相違を明確にするため，SCIP による計算時間を横軸に取り，対応するインスタンスの FiberSCIP の加速率 (SCIP による計算時間/FiberSCIP による計算時間) を縦軸に取ったグラフを図 4 に示す．図 4 より，1 回の前処理のケースが，SCIP とほとんど同じ振る舞いをするのに対して，2 回の前処理を行った結果の振る舞いはインスタンスによって大きく異なり，2 回の前処理のオーバーヘッドはあるが，それが必ずしも減速に繋がるわけではなく，加速するケースもかなりあることがわかる．2 回の前処理を行うということ

表 2　Sequential executions of SCIP and FiberSCIP

Name	SCIP			FiberSCIP (Presolve Once)			FiberSCIP (Presolve Twice)			Solution Status
	Nodes	Time	Gap	Nodes	Time	Gap	Nodes	Time	Gap	
30n20b8	4691	> 7201	267.5	8753	> 7202	166.2	4639	> 7219	200.0	ok
acc-tight5	1649	340	0.0	1649	339	0.0	1808	353	0.0	ok
aflow40b	286358	2967	0.0	286358	2971	0.0	668667	6444	0.0	ok
air04	144	126	0.0	144	127	0.0	215	149	0.0	ok
app1-2	76	1997	0.0	76	2004	0.0	392	2163	0.0	ok
ash608gpia-3col	7	77	–	7	80	–	7	77	–	ok
bab5	17595	> 7202	1.1	17615	> 7202	1.1	12846	> 7201	1.3	ok
beasleyC3	1151024	> 7203	25.7	1193762	> 7203	25.7	992738	> 7203	13.9	ok
biella1	3270	1549	0.0	3270	1552	0.0	3270	1554	0.0	ok
bienst2	117922	553	0.0	117922	553	0.0	86468	439	0.0	ok
binkar10_1	114820	257	0.0	114820	256	0.0	135435	280	0.0	ok
bley_xl1	17	467	0.0	17	449	0.0	1	342	0.0	ok
bnatt350	6685	1121	0.0	6685	1130	0.0	4134	859	0.0	ok
core2536-691	281	1064	0.0	281	1066	0.0	810	1487	0.0	ok
cov1075	965740	> 7201	8.5	953142	> 7201	8.5	976932	> 7201	8.5	ok
csched010	651293	> 7200	4.3	646222	> 7201	4.4	684362	> 7200	4.7	ok
danoint	881262	> 7200	2.7	879302	> 7201	2.7	870942	> 7201	2.6	ok
dfn-gwin-UUM	87889	239	0.0	87889	241	0.0	87613	239	0.0	ok
eil33-2	10571	180	0.0	10571	180	0.0	10571	190	0.0	ok
eilB101	9239	1287	0.0	9239	1300	0.0	9239	1292	0.0	ok
enlight13	1116961	1388	0.0	1116961	1355	0.0	745476	857	0.0	ok
enlight14	172555	229	–	172555	211	–	259155	316	–	ok
ex9	1	191	0.0	1	191	0.0	1	194	0.0	ok
glass4	1	! 5244	–	1	! 5150	–	1	! 4138	–	abort
gmu-35-40	1	! 1905	–	1	! 1849	–	1	! 1785	–	abort
iis-100-0-cov	106874	2685	0.0	106874	2694	0.0	106874	2669	0.0	ok
iis-bupa-cov	111227	> 7200	8.7	113790	> 7201	8.6	113309	> 7202	8.6	ok
iis-pima-cov	13011	1504	0.0	13011	1505	0.0	12469	1526	0.0	ok
lectsched-4-obj	2772	156	0.0	2772	157	0.0	21026	826	0.0	ok
m100n500k4r1	5025592	> 7201	4.0	5072925	> 7200	4.0	5050160	> 7201	4.0	ok
macrophage	528171	> 7200	36.6	528062	> 7200	36.6	541612	> 7201	37.7	ok
map18	293	862	0.0	293	857	0.0	261	1182	0.0	ok
map20	353	1001	0.0	353	1010	0.0	526	926	0.0	ok
mcsched	16113	339	0.0	16113	339	0.0	16113	341	0.0	ok
mik-250-1-100-1	1421995	474	0.0	1421995	457	0.0	1421995	457	0.0	ok
mine-166-5	6949	104	0.0	6949	105	0.0	6170	109	0.0	ok
mine-90-10	92852	543	0.0	92852	541	0.0	189285	1364	0.0	ok
msc98-ip	626	> 7203	–	624	> 7217	100.0	485	> 7212	57.0	ok
mspp16	1	! 144	–	1	! 143	–	1	! 144	–	abort
mzzv11	2621	755	0.0	2621	760	0.0	8361	1868	0.0	ok
n3div36	1	! 3101	–	1	! 2857	–	1	! 2825	–	abort
n3seq24	542	> 7228	16.5	1	! 71	–	1	! 72	–	abort
n4-3	48686	1281	0.0	48686	1285	0.0	57354	1466	0.0	ok
neos-1109824	24162	356	0.0	24162	359	0.0	20142	283	0.0	ok
neos-1337307	238371	> 7200	0.0	238721	> 7200	0.0	239423	> 7201	0.0	ok
neos-1396125	65896	6547	0.0	65896	6563	0.0	40087	4125	0.0	ok
neos13	17151	> 7201	25.3	17220	> 7201	25.3	17847	> 7201	25.3	ok
neos-1601936	19862	> 7200	200.0	19874	> 7200	200.0	7771	5311	0.0	ok
neos18	9133	63	0.0	9133	65	0.0	6827	81	0.0	ok
neos-476283	1	! 483	–	1	! 163	–	1	! 273	–	abort
neos-686190	9894	199	0.0	9894	200	0.0	4739	134	0.0	ok
neos-849702	50303	1396	0.0	50303	1396	0.0	126716	3193	0.0	ok
neos-916792	57471	449	0.0	57471	450	0.0	66213	511	0.0	ok
neos-934278	2526	> 7200	1.7	2567	> 7201	1.7	2755	> 7206	1.0	ok
net12	7260	5950	0.0	7260	5938	0.0	3678	4536	0.0	ok
netdiversion	33	> 7278	5.2	1	> 7511	0.0	26	> 7295	100.0	ok
newdano	3062795	> 7200	7.7	3052375	> 7201	7.7	2842106	> 7201	7.7	ok
noswot	463569	197	0.0	463569	191	0.0	575618	217	0.0	ok
ns1208400	12202	3251	0.0	12202	3252	0.0	4817	1361	0.0	ok
ns1688347	4995	510	0.0	4995	510	0.0	4538	528	0.0	ok
ns1758913	1	! 3043	–	1	! 2853	–	1	! 3132	–	abort
ns1766074	946987	1051	–	946987	959	–	945109	1318	–	ok
ns1830653	42213	746	0.0	42213	745	0.0	88276	1111	0.0	ok
opm2-z7-s2	4401	2345	0.0	4401	2357	0.0	1978	1142	0.0	ok
pg5_34	318742	2396	0.0	318742	2387	0.0	318742	2339	0.0	ok
pigeon-10	3757759	> 7200	10.0	3535086	> 7200	10.0	2955734	> 7200	10.0	ok
pw-myciel4	494676	> 7200	11.1	482684	> 7200	11.1	488828	> 7201	25.0	ok
qiu	11012	86	0.0	11012	87	0.0	9811	83	0.0	ok
rail507	1472	2772	0.0	1472	2868	0.0	1319	2126	0.0	ok
ran16x16	371304	335	0.0	371304	332	0.0	441109	396	0.0	ok
reblock67	140595	535	0.0	140595	533	0.0	83043	339	0.0	ok
rmatr100-p10	901	297	0.0	901	297	0.0	901	298	0.0	ok
rmatr100-p5	397	857	0.0	397	857	0.0	397	855	0.0	ok
rmine6	530871	4728	0.0	530871	4712	0.0	687364	6508	0.0	ok
rocII-4-11	13591	406	0.0	13591	405	0.0	13194	399	0.0	ok
rococoC10-001000	365708	2679	0.0	365708	2687	0.0	1075742	> 7201	1.4	ok
roll3000	977873	> 7200	1.4	978756	> 7201	1.4	1322672	> 7201	0.9	ok
satellites1-25	9374	3902	0.0	9374	3879	0.0	16711	4956	0.0	ok
sp98ic	1	! 5253	–	1	! 5110	–	1	! 5053	–	abort
sp98ir	4807	120	0.0	4807	123	0.0	8984	202	0.0	ok
tanglegram1	27	1808	0.0	27	1815	0.0	27	2358	0.0	ok
tanglegram2	3	11	0.0	3	12	0.0	3	13	0.0	ok
timtab1	1039843	744	0.0	1085175	768	0.0	1014862	746	0.0	ok
triptim1	30	4535	0.0	12	5122	0.0	39	> 7207	0.0	ok
unitcal_7	17499	3734	0.0	17499	3731	0.0	12785	2501	0.0	ok
vpphard	2319	> 7203	–	2318	> 7203	–	3693	> 7206	–	ok
zib54-UUE	736111	> 7200	8.1	553363	6934	0.0	565989	6971	0.0	ok
solved/timeout/abort		58/22/7			59/20/8			58/21/8		
all solved: 56/87 geom. mean	8711	623.5		8718	623.7		9204	669.5		
all solved: 58/87 geom. mean	8425	661.7		8299	663.3					
all solved: 57/87 geom. mean				9376	650.6		9894	697.6		
geom. mean (comp.)		1466.2			1467.9			1554.7		

は，1 回の前処理とはソルバのアルゴリズムが変わったことを意味し，また，2 回の前処理を FiberSCIP のデフォルト設定としている．よって，2 回の前処理を行った結果を，並列化の効果を調べるための基準となる逐次処理の結果として採用した．

次に，予備実験として，制限時間内 Racing 段階だけを実行し，Solver 間での上界値の通信を行わない設定で，Solver スレッド数を変化させた FiberSCIP の数値実験を行った．つまり，Solver 数分の異なるパラメタ設定で，それぞれ Solver を実行し，最も速く終了した Solver の実行時間を計算終了の時間とする．表 3 は，Solver スレッド数を，2, 4, 6, 8 と増やして行った数値実験結果である．表 2 の値に加えて，加速率と，基準としている 1 Solver スレッドの結果と比較した，解けたインスタンス数の差が示されている．Solver 数を一つ増やすと一つ多くのパラメタ設定を試す Solver が追加され，上界値の通信はないので，論理的には Solver 数を増やすほど速くなる (上界値の通信を行うと一般的には加速するが，受け取った上界値が Solver に設定されることで，より良い実

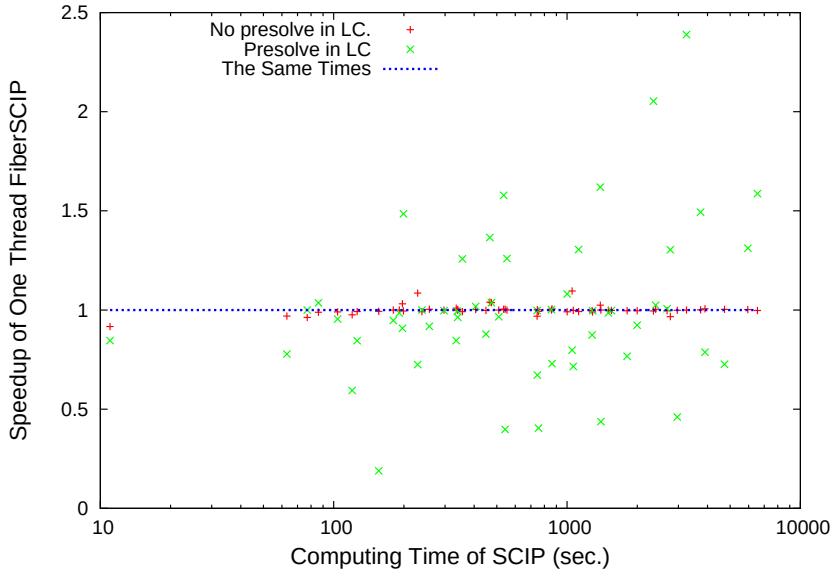

図 4　Relative computing time ratio of FiberSCIP compared to SCIP

行可能解の発見が遅れることがあり，Solver 数を増やすほど速くなるとは論理的にも言えない)．しかし実際には，Solver 数が増えると，各 Solver がメモリへアクセスする速度は，キャッシュミスやメモリアクセス時のバスの競合などにより低下し，1 Solver 当たりの性能が低下する．実際，表 3 が示すように，最も良い性能を示しているのは，4 Solver スレッドでの結果であり，8 コアのコンピュータを使用しディスクアクセスが生じないように SCIP のパラメタを設定しているにも関わらず，6 Solver スレッド では既に性能低下しており，8 Solver スレッドでの計算時間の幾何平均値は逐次処理よりも遅く，制限時間内に解けるインスタンス数も逐次処理より少ない．

1 Solver 当たりの性能低下を調べるため，表 3 のインスタンス名に (i) と (i*) の印をつけた結果を詳細に調べた．(i) は，1, 2, 4, 6 Solver スレッド数で同じ数の部分問題を解いたインスタンスであり，勝者 Solver が同じであると考えられる．一方，(i*) は，6 Solver スレッドだけが異なる部分問題数であり，かなり性能の近い競合する複数の Solver の存在を示唆する．競合する複数の Solver が存在するために 8 Solver による結果だけが異なるケースがあるかもしれないが，その理由で異なるのかどうかの識別ができないため対象から外した．(i) および (i*) のインスタンスに対する結果について，横軸に定常的なメモリ容量の見積もりとしての (変数の数×制約式の数) を取り，縦軸に加速率をプロットしたグラフが図 5 である．現在のコンピュータアーキテクチャの内部構造は，簡単なモデルで説明できるほど単純ではない．しかし，簡易に入手できる情報だけから予測したいという要求はある．そこで，x を定常的なメモリ利用量として $f(x) = 1.0 - c\log(x)$ の割合で 1 Solver スレッド当たりの性能は低下するという仮定で，スレッド数ごとにあてはめを行い，その結果も図 5 に示した．メモリを全く使用しないような計算においては性能低下は起こらないという仮定と，メモリ不足などで異常終了せず，計算が実行される限り漸近的にある値に近づくという仮定のもと $f(x)$ を設定した．かなりの上下変動は存在するが，全体としての挙動は，ある程度 $f(x)$ と合致している．係数 c の値は，同時に動作させる Solver スレッドの数が増えれば大きくなる傾向にあり，この値は利用するコンピュータ環境に依存すると考えられる．FiberSCIP では，スレッド数を増やすと，基本的には 1 Solver スレッド当たりの速度低下を生じるため，実装されているコア数を全て動作させる場合には，並列探索の効果が，このような速度低下を上回る必要がある．

次の予備実験として，先の予備実験と同様であるが，Solver 間で見つかった上界値の通信を行い，それにより限定操作を働かせた場合の効果について調べた．上界値の通信を行わない場合，計算終了時間が極めて近い Solver スレッドが存在するときには，非決定的な振る舞いはあるが，その例を除くと，非決定的な振る舞いはない．よって，先

表 3　Only racing stage runs without upper bounds communications

Name	One thread		2 threads		4 threads		6 threads		8 threads	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
30n20b8	4639	> 7219	2539	> 7212	1	> 7200	2038	> 7207	5277	> 7207
acc-tight5	1808	353	1808	353	777	289	748	189	308	156
aflow40b	668667	6444	238236	2080	194236	1818	192294	1987	178690	2220
air04	215	149	179	140	101	101	101	110	101	129
app1-2	392	2163	392	2192	47	1716	47	2776	47	3991
ash608gpia-3col (i)	7	77	7	79	7	102	7	174	7	248
bab5	12846	> 7201	12206	> 7210	1	> 7201	1	> 7201	1	> 7203
beasleyC3	992738	> 7203	1	> 7200	1	> 7200	1	> 7200	1	> 7200
biella1 (i)	3270	1554	3270	1603	3270	1614	3270	1725	3270	2332
bienst2	86468	439	86468	447	86468	450	86468	468	84244	460
binkar10_1	135435	280	135435	294	135435	306	67149	361	67149	432
bley_xl1 (i)	1	342	1	350	1	349	1	352	1	361
bnatt350	4134	859	4134	866	5058	864	4134	919	2831	945
core2536-691	810	1487	192	1078	192	1096	192	1361	192	2566
cov1075	976932	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
csched010	684362	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
danoint	870942	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
dfn-gwin-UUM	87613	239	87613	242	63562	198	43321	209	22548	218
eil33-2 (i*)	10571	190	10571	208	10571	228	11423	291	10571	380
eilB101	9239	1292	10512	1124	9770	989	9770	1039	7381	805
enlight13	745476	857	745476	878	367322	510	109200	839	109200	418
enlight14	259155	316	259155	319	259155	388	259155	518	100054	311
ex9 (i)	1	194	1	188	1	183	1	191	1	191
glass4	1	! 4138	1	! 6086	1082160	1655	1	! 1510	1082160	2281
gmu-35-40	1	! 1785	1	! 2376	1	! 3117	1	! 3486	1	! 3988
iis-100-0-cov (i)	106874	2669	106874	2696	106874	2769	106874	3201	106874	3293
iis-bupa-cov	113309	> 7202	1	> 7200	1	> 7200	1	> 7200	1	> 7200
iis-pima-cov	12469	1526	12469	1493	6523	1318	6523	1394	6523	2368
lectsched-4-obj	21026	826	4591	587	3213	375	1796	208	1796	265
m100n500k4r1	5050160	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
macrophage	541612	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
map18	261	1182	261	1589	493	1418	493	2129	347	3595
map20	526	926	526	958	399	1085	295	1707	333	2148
mcsched	16113	341	16113	342	16625	343	14860	339	13508	319
mik-250-1-100-1 (i)	1421995	457	1421995	468	1421995	505	1421995	529	1421995	551
mine-166-5	6170	109	6170	117	1583	91	4289	101	3582	126
mine-90-10	189285	1364	189285	1376	74459	826	74459	1071	74459	1374
msc98-ip	485	> 7212	393	> 7230	125	> 7209	102	> 7221	1	> 7208
mspp16	1	! 144	1	! 159	1	! 2541	1	! 2430	1	! 2566
mzzv11	8361	1868	4756	1954	4756	1784	4877	2355	4665	4148
n3div36	1	! 2825	1	! 3386	1	! 4023	1	> 7201	1	> 7201
n3seq24	1	! 72	1	! 721	1	> 7225	6	> 7258	4	> 7312
n4-3	57354	1466	57354	1543	57354	1554	43434	1469	43434	1677
neos-1109824	20142	283	8268	135	8268	169	8268	217	8268	355
neos-1337307	239423	> 7201	182852	> 7200	1	> 7200	1	> 7200	1	> 7200
neos-1396125	40087	4125	40087	4131	40087	4164	38909	4204	38909	4258
neos13	17847	> 7201	1	> 7202	1	> 7202	1	> 7201	1	> 7201
neos-1601936	7771	5311	7771	5323	7771	5468	9458	5655	9458	6936
neos18	6827	81	6827	80	7855	76	5830	60	5830	73
neos-476283	1	! 273	372	1017	1	! 876	372	1667	372	2632
neos-686190	4739	134	4739	134	4739	151	3241	133	3241	164
neos-849702	126716	3193	20907	774	20907	786	17371	822	20907	803
neos-916792 (i*)	66213	511	66213	518	66213	606	60176	796	66213	933
neos-934278	2755	> 7206	248	> 7218	161	> 7216	139	> 7207	45	> 7229
net12	3678	4536	3513	4152	3513	4162	3513	6633	1	> 7201
netdiversion	26	> 7295	24	> 7313	1	> 7316	1	> 7718	1	> 7371
newdano	2842106	> 7201	1	> 7200	2874552	7199	2058508	6520	2058508	7024
noswot	575618	217	575618	225	434470	188	393317	225	324848	260
ns1208400	4817	1361	691	491	691	504	691	550	691	534
ns1688347 (i)	4538	528	4538	534	4538	536	4538	547	4538	602
ns1758913	1	! 3132	1	! 3903	1	! 4869	-	-	2	> 7881
ns1766074	945109	1318	907103	1367	934754	959	934754	1441	934754	1871
ns1830653	88276	1111	88276	1107	61053	950	61053	1149	60186	1178
opm2-z7-s2 (i*)	1978	1142	1978	1297	1978	2238	1822	2405	1978	5401
pg5_34	318742	2339	318742	2407	318742	2782	318742	3239	253046	4173
pigeon-10	2955734	> 7200	1	> 7200	1	> 7200	1	! 613	1	! 3071
pw-myciel4	488828	> 7201	1	> 7201	115605	4945	296556	4496	296556	4666
qiu (i)	9811	83	9811	83	9811	84	9811	87	9811	91
rail507	1319	2126	1319	3837	1319	4127	1319	4578	1	> 7202
ran16x16	441109	396	441109	401	356748	350	356748	401	356748	499
reblock67 (i)	83043	339	83043	344	83043	335	83043	412	83043	482
rmatr100-p10 (i)	901	298	901	298	901	299	901	325	901	415
rmatr100-p5 (i*)	397	855	397	859	397	858	413	905	397	2171
rmine6	687364	6508	687364	6650	1	> 7200	1	> 7200	1	> 7200
rocII-4-11	13194	399	13194	398	11288	342	11288	354	11288	443
rococoC10-001000	1075742	> 7201	1	> 7200	562119	4846	562119	5755	1	! 4231
roll3000	1322672	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
satellites1-25	16711	4956	16711	5064	3656	4244	3656	6115	11313	> 7203
sp98ic	1	! 5053	1	! 5845	1	! 6251	1	> 7201	1	> 7201
sp98ir	8984	202	7236	199	8685	215	4999	183	4999	223
tanglegram1	27	2358	27	2516	27	3941	27	4546	22	> 7204
tanglegram2 (i)	3	13	3	13	3	13	3	18	3	20
timtab1 (i)	1014862	746	1014862	768	1014862	773	1014862	955	1014862	1027
triptim1	39	> 7207	162	6881	1	> 7346	1	> 7300	9	3183
unitcal_7	12785	2501	12785	2529	12785	3906	1	> 7202	1	> 7201
vpphard	3693	> 7206	5	> 7210	2	> 7226	1	> 7202	2	> 7215
zib54-UUE	565989	6971	565989	7090	1	> 7200	1	> 7200	1	> 7200
solved/timeout/abort	58/21/8		60/20/7		60/21/6		59/23/5		56/27/4	
all solved: 51/87 geom. mean	9817	573.9	8148	529.7	6935	507.5	6386	569.4	5905	672.0
speedup				1.084		1.131		1.008		0.854
diff. # of solved				+2		+2		+1		-2
geom. mean (comp.)		1554.6		1460.1		1434.4		1564.4		1723.6
all solved: 51/87 geom. mean			8148	529.7	6935	507.5	6386	569.4	5905	672.0

の予備実験は，各設定で 1 回の試行結果を示した．一方，Solver 間で上界値の通信を行うと，それが伝達されるタイミングにより，同じ動作環境，同じパラメタ設定でも，結果が大きく異なる非決定的な振る舞いを示す．従って，この予備実験は，5 回の試行による平均値により評価する．数値実験は先の予備実験で最も良い結果を得た 4 Solver スレッドと，実装されているコアを全て使うことを意図した 8 Solver スレッドにより行った．4 Solver スレッドの結果を表 4，8 Solver スレッドの結果を表 5 に示す．ここでは，上界値の通信による効果を調べたいので，比較対象として上界値の通信を行わない結果を再度示し，続いて 5 回の試行の結果が示されている．実際，同じパラメタ設定，同じ Solver スレッド数でも結果が大きく異なるインスタンスがある．左端のカラムには，加速率と 5 回の各試行において解けたインスタンス数の平均が示されており，4 Solver スレッドでは，平均で 5% 程度加速し，制限時間内に解けるインスタンス数も平均 2 問程度増えている．一方，8 Solver スレッドでは，加速率は，4 Solver スレッドの場合より大きく，16% 程度加速し，制限時間内に解けるインスタンス数は平均 1 問程度増えている．並列

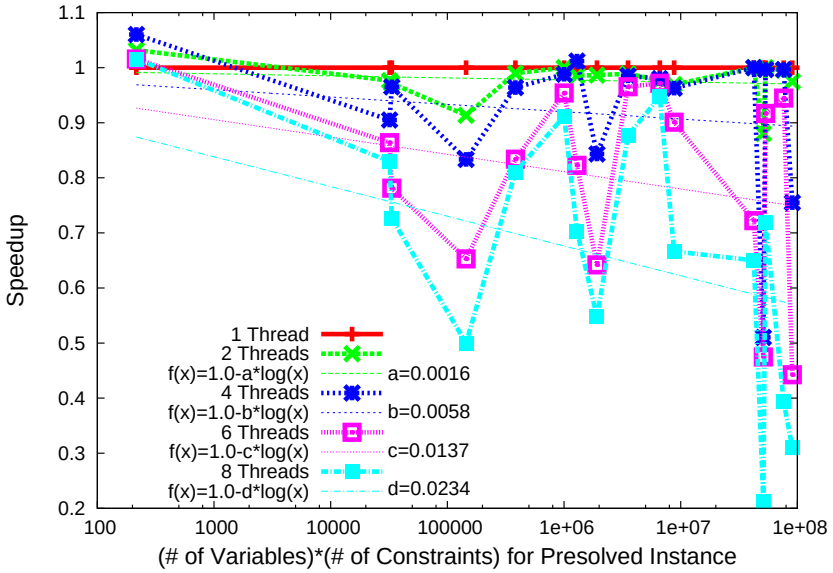

図 5　Relative computing time ratio of FiberSCIP for each number of SOLVER threads used

探索では，早期に良い実行可能解を発見することが多いため，早期に発見された上界値による限定操作の効果と考えられる．ただし，逐次処理と比較すると，8 SOLVER スレッドでの性能は表 3 では 15% 低下しているので，上界値通信による Racing 段階のみの実行性能は逐次処理の結果と同程度となる．

これまでの予備実験の結果を踏まえて，Racing Ramp-Up と Normal Ramp-Up の性能比較と，2 回の前処理を実行することの効果について調べた．全てのインスタンスの結果を掲載すると多大なページ数を必要とするので，解けたインスタンス数と加速率をまとめた表 6 を示す．表 6 の全ての結果は，2 回の前処理を実行する逐次処理の結果と比較している．n SOLVER スレッドによる加速率は，

$$\text{Speedup}(n) = \frac{\text{FiberSCIP 逐次処理の計算時間の幾何平均}}{\text{FiberSCIP } n \text{ SOLVER スレッドによる計算時間の幾何平均}}$$

として求めているが，幾何平均の対象は，各パラメタ設定において全ての試行で解けたインスタンスの計算時間であり，異なるパラメタ設定における結果間で比較すると，解けたインスタンスは異なり，解けたインスタンス数も異なる．左端 Settings のカラムに，基準となる逐次処理の結果を含む全ての試行において解けたインスタンス数が示されている．この数のインスタンスが，加速率の計算に用いられた．それと同時に，基準となる逐次処理を除く全ての試行で解けたインスタンス数が ( ) 内に示されている．基準となる逐次処理において解けたインスタンス数は 58 である．この値を考慮すると，逐次処理では解けたが並列処理を適用すると解けなかったインスタンスが存在する一方，( ) 内の値は多くのケースで 58 を超えており，逐次処理では解けないが，並列処理では全ての試行において定常的に解けるインスタンスもあったことがわかる．

表 6 の結果をもとに考察を行う．まず，2 回の前処理を実行することの効果であるが，LOADCOORDINATOR において前処理を行わない場合 (no presolve) に，計算時間の観点から性能を落としていることがわかる．この場合，加速率の計算には，1 回だけ前処理を行う逐次処理の結果を基準として利用した方が好ましいが，ここでは他と共通して比較するために，2 回の前処理による結果と比較している．前処理の回数にかかわらず，逐次処理を含む数値実験では，計算対象となるインスタンス数は少なくなる傾向がある．そこで，逐次処理を除いて解けたインスタンス数を示す ( ) 内の値を見ると，5 回の全試行により解けたインスタンス数という観点においても性能が落ちていることがわかる．この結果は，LOADCOORDINATOR において前処理を行わない場合，各 SOLVER が定常的に持つインスタンスデータの量が増えることによって，SOLVER 当たりの性能も低下するので，予想された通りである．したがって，

表 4　Effects of upper bounds communication (4 Solver Threads)

	Without comm.		Run - 1		Run - 2		Run - 3		Run - 4		Run - 5	
Name	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
30n20b8	1	> 7200	902	> 7218	1022	> 7209	820	> 7204	516	> 7203	1	> 7202
acc-tight5	777	289	39	62	39	62	39	61	39	61	39	61
aflow40b	194236	1818	181535	1875	181535	1856	181535	1841	175610	1639	198942	1611
air04	101	101	49	97	49	96	49	96	49	96	49	96
app1-2	47	1716	47	1930	47	1893	47	1953	47	1902	47	1923
ash608gpia-3col	7	102	7	101	7	100	7	101	7	99	7	100
bab5	1	> 7201	1	> 7201	15454	> 7202	1	> 7201	1	> 7201	1	> 7201
beasleyC3	1	> 7200	1	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
biella1	3270	1614	2288	1159	2654	1340	1649	1408	1658	1062	1575	1309
bienst2	86468	450	94998	465	97769	499	104468	506	101544	531	104990	523
binkar10_1	135435	306	165250	395	136563	343	107798	259	121683	292	111086	267
bley_xl1	1	349	1	349	1	349	1	351	1	352	1	338
bnatt350	5058	864	5058	864	5058	863	5058	814	5058	864	5058	861
core2536-691	192	1096	901	1335	901	1337	810	1607	901	1340	810	1575
cov1075	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
csched010	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
danoint	1	> 7200	704140	6653	704140	6662	704140	6619	704140	6684	704140	6643
dfn-gwin-UUM	63562	198	63710	199	63909	201	63551	201	64307	201	61020	186
eil33-2	10571	228	9879	209	9401	201	8975	197	9759	212	9772	208
eilB101	9770	989	4872	692	4402	562	5689	758	4811	624	5049	659
enlight13	367322	510	367322	395	367322	490	367322	489	367322	496	367322	489
enlight14	259155	388	259155	305	259155	388	259155	556	259155	390	259155	398
ex9	1	183	1	191	1	197	1	192	1	191	1	191
glass4	1082160	1655	939959	1544	702114	1167	1050611	1650	2542544	3495	3421535	4949
gmu-35-40	1	! 3117	1	! 2458	1	! 2144	1	! 2429	1	! 2235	1	! 2415
iis-100-0-cov	106874	2769	98732	2731	94430	2598	104420	2863	91513	2490	104420	2855
iis-bupa-cov	1	> 7200	1	> 7200	1	> 7200	1	> 7200	106616	> 7200	1	> 7200
iis-pima-cov	6523	1318	7124	1146	7124	1149	7124	1148	7124	1144	7124	1145
lectsched-4-obj	3213	375	4540	264	337	184	3506	237	2794	215	563	243
m100n500k4r1	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
macrophage	1	> 7200	1	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
map18	493	1418	493	1427	493	1403	493	1428	493	1465	493	1410
map20	399	1085	399	1180	399	1097	399	1567	399	1184	399	1101
mcsched	16625	343	13704	309	20407	406	16861	344	15775	336	14646	312
mik-250-1-100-1	1421995	505	1421995	490	1421995	495	1421995	500	1421995	496	1421995	494
mine-166-5	1583	91	5138	85	3576	83	4167	87	1028	84	2888	82
mine-90-10	74459	826	291363	2503	65635	494	199943	1949	96075	873	199321	1483
msc98-ip	125	> 7209	45	> 7242	138	> 7209	1	> 7201	130	> 7242	138	> 7220
mspp16	1	! 2541	1	! 2544	1	! 2551	1	! 2536	1	! 2562	1	! 2575
mzzv11	4756	1784	4048	1147	6993	1612	4461	1264	4806	1305	4152	1227
n3div36	1	! 4023	1	! 4683	1	! 5047	1	! 4698	1	! 4788	1	! 4707
n3seq24	1	> 7225	1	> 7209	1	> 7210	285	> 7242	156	> 7225	1	! 674
n4-3	57354	1554	49902	1301	51789	1403	46044	1293	48271	1355	50043	1406
neos-1109824	8268	169	8164	171	8164	285	8164	171	8164	171	8164	175
neos-1337307	1	> 7200	1	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
neos-1396125	40087	4164	39799	4152	42121	4796	41808	4444	44725	4925	43032	4750
neos13	1	> 7202	4882	3741	50	2952	46	3674	6882	4596	45	2895
neos-1601936	7771	5468	12605	> 7200	3967	3662	6206	6749	1	> 7200	3191	3404
neos18	7855	76	6292	70	6292	70	6292	70	6292	71	6292	70
neos-476283	1	! 876	1	! 439	1	! 2840	1	! 869	1	! 855	1	! 1117
neos-686190	4739	151	3589	118	4306	129	3018	109	3679	123	3970	124
neos-849702	20907	786	20907	788	20907	786	20907	790	20907	788	20907	788
neos-916792	66213	606	66296	584	66044	581	68269	836	64479	594	69786	641
neos-934278	161	> 7216	712	> 7206	207	> 7205	171	> 7201	291	> 7222	209	> 7202
net12	3513	4162	3850	3314	3056	4018	3484	4383	4464	4095	4721	4216
netdiversion	1	> 7316	15	> 7315	1	> 7353	-	-	1	> 7317	1	> 7320
newdano	2874552	7199	2870908	6711	1	> 7200	1898642	4595	2394587	7152	2791732	6957
noswot	434470	188	301829	159	301829	158	301829	164	301829	170	301829	163
ns1208400	691	504	691	506	691	507	691	507	691	507	691	510
ns1688347	4538	536	3515	483	3752	476	4538	536	4310	505	2944	383
ns1758913	1	! 4869	1	! 7898	1	! 5011	1	! 4796	1	! 4709	-	-
ns1766074	934754	959	934754	1194	934754	1190	934754	1188	934754	1170	934754	1180
ns1830653	61053	950	38073	677	48232	810	43242	748	39359	655	37962	679
opm2-z7-s2	1978	2238	3233	2303	2321	2693	2156	4070	2485	2744	2285	2416
pg5_34	318742	2782	292798	2336	318401	2726	310197	2319	309644	2589	267308	1752
pigeon-10	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
pw-myciel4	115605	4945	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
qiu	9811	84	9543	87	9471	85	9467	86	9483	86	9471	85
rail507	1319	4127	1303	4485	1334	4519	1191	3921	1213	4174	1329	3920
ran16x16	356748	350	365028	380	392537	365	365185	371	377618	385	361003	367
reblock67	83043	335	107761	475	107282	438	91057	555	110107	445	107846	610
rmatr100-p10	901	299	901	299	901	309	901	307	901	306	901	308
rmatr100-p5	397	858	397	881	397	883	397	885	397	882	397	882
rmine6	1	> 7200	1	> 7200	563296	5305	561803	6544	578703	6873	607822	7126
rocII-4-11	11288	342	16131	413	14119	388	12484	348	10857	328	9250	306
rococoC10-001000	562119	4846	650113	4766	510849	4016	359182	3362	646531	5183	545121	4990
roll3000	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
satellites1-25	3656	4244	11745	4256	11745	4124	11745	4289	11745	4309	11745	4354
sp98ic	1	! 6251	1	> 7201	1	> 7201	1	> 7200	1	> 7200	1	> 7201
sp98ir	8685	215	5720	142	4831	132	5312	130	5361	128	5902	140
tanglegram1	27	3941	27	4031	27	3914	27	3982	27	3991	27	5135
tanglegram2	3	13	3	14	3	13	3	13	3	13	3	14
timtab1	1014862	773	781984	715	758833	622	744540	609	708098	650	697892	573
triptim1	1	> 7346	3	> 7266	3	> 7233	3	> 7246	4	> 7226	3	> 7261
unitcal_7	12785	3906	15221	4499	15613	4424	12599	3796	12996	3933	16640	3570
vpphard	2	> 7226	2	> 7207	1	> 7201	2	> 7205	2	> 7203	2	> 7205
zib54-UUE	1	> 7200	547411	7111	564644	7073	546678	7018	546544	7015	570034	7021
solved/timeout/abort	60/21/6		61/21/5		62/20/5		63/18/5		62/20/5		63/18/5	
all solved: 57/87												
geom. mean: 591.69	7110	620.7	7047	591.0	6526	581.2	6715	604.8	6678	587.2	6661	594.3
speedup: 1.049				1.05		1.07		1.03		1.06		1.04
diff. # of solved: +2.2				+1		+2		+3		+2		+3
geom. mean (comp.): 1379.54		1434.4		1386.5		1353.3		1398.4		1383.9		1375.7
all solved: 60/87												
geom. mean: 661.36			8132	661.5	7008	648.3	7186	675.7	7770	659.5	7134	661.9

FiberSCIP においては，前処理後インスタンスを転送するというデフォルト設定は妥当であると考えられる．

次に，Normal Ramp-Up と Racing Ramp-Up の結果について考察する．表 6 から，4 Solver スレッドに対しては，加速率や解けたインスタンス数の観点から Normal Ramp-Up の方が優位と考えられる．しかし，加速率計算の対象としているインスタンス数は，Racing Ramp-Up の方が多い．これは，安定して常に解けるインスタンス数が多いことを意味する．この傾向は，逐次処理の結果を除いた場合でも同様である．4 Solver しか利用しない場合には，Racing 段階による動的なパラメタ調整の優位性が出ず，最初から並列探索を行った方が加速しているとみなせる．それでも，Racing Ramp-Up は，Racing 段階の結果により動的に挙動を変えることで，より多くのインスタンスを安定的に解いているという意味では優位性がある．8 Solver スレッドにおいては，Racing 段階でのパラメタ調整の効果が顕著になり，制限時間内に解けるインスタンス数に関しては，Racing Ramp-Up は Normal Ramp-Up と比較して相対的に優位性を増している．加速率は，Normal Ramp-Up の方が良いが，Racing Ramp-Up の方が

表 5　Effects of upper bounds communication (8 SOLVER Threads)

Name	Without comm.		Run - 1		Run - 2		Run - 3		Run - 4		Run - 5	
	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
30n20b8	5277	> 7207	6696	> 7204	1	> 7201	7233	> 7204	6775	> 7216	7004	> 7217
acc-tight5	308	156	39	69	39	68	39	75	39	63	39	69
aflow40b	178690	2220	146859	1650	145062	1816	143459	1775	142275	1730	136915	1755
air04	101	129	41	119	44	120	59	119	94	99	44	118
app1-2	47	3991	254	3847	47	3905	47	2439	47	2478	47	3831
ash608gpia-3col	7	248	7	249	7	254	7	475	7	259	7	247
bab5	1	> 7203	1	> 7201	1	> 7202	1	> 7201	1	> 7202	1	> 7202
beasleyC3	1	> 7200	1	> 7201	1	> 7200	1	> 7200	1	> 7200	1	> 7200
biella1	3270	2332	1734	1148	1721	1741	1605	1144	2079	1426	2564	1048
bienst2	84244	460	91266	539	84210	454	87102	530	81682	431	81735	504
binkar10_1	67149	432	119952	371	142863	531	141806	489	129169	459	119952	536
bley_xl1	1	361	1	361	1	361	1	401	1	364	1	363
bnatt350	2831	945	2831	980	2831	693	2831	1533	2831	999	2831	984
core2536-691	192	2566	101	1318	325	2134	101	1298	87	1340	49	1258
cov1075	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
csched010	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
danoint	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
dfn-gwin-UUM	22548	218	20837	125	19914	107	42928	182	20837	123	58127	224
eil33-2	10571	380	10228	350	10508	366	8207	344	9958	359	8561	335
eilB101	7381	805	7897	881	8227	682	5314	702	5593	565	3626	647
enlight13	109200	418	138288	335	125558	201	188996	409	153848	377	167082	434
enlight14	100054	311	100054	298	100054	300	100054	309	100054	292	1	! 304
ex9	1	191	1	193	1	191	1	189	1	190	1	191
glass4	1082160	2281	1323490	5001	507573	932	709838	1807	1748611	4002	450004	1229
gmu-35-40	1	! 3988	1	! 2505	1	! 2687	1	! 2772	1	! 2430	1	! 2776
iis-100-0-cov	106874	3293	99765	3265	104531	3438	97503	3306	93341	3249	93341	3275
iis-bupa-cov	1	> 7200	1	> 7200	1	> 7199	1	> 7200	1	> 7200	1	> 7200
iis-pima-cov	6523	2368	6547	2021	7155	2058	7589	2076	7321	2810	7321	2060
lectsched-4-obj	1796	265	1184	131	672	108	340	109	771	102	162	113
m100n500k4r1	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
macrophage	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
map18	347	3595	347	3421	339	3516	347	3571	347	3438	347	3455
map20	333	2148	333	2204	333	2262	333	2363	333	2172	333	1117
mcsched	13508	319	15356	348	12850	345	15074	355	12239	302	12840	340
mik-250-1-100-1	1421995	551	1421995	565	1432187	519	1421995	575	1421995	565	1421995	571
mine-166-5	3582	126	1792	97	2399	112	1848	100	1494	96	3028	112
mine-90-10	74459	1374	96792	1170	101568	1194	60457	719	98243	749	167794	2074
msc98-ip	1	> 7208	24	> 7222	24	> 7245	23	> 7215	24	> 7224	23	> 7212
mspp16	1	! 2566	1	! 2561	1	! 2413	1	! 2574	1	! 2569	1	! 2570
mzzv11	4665	4148	4374	2523	5502	2496	4131	2616	8043	3118	4931	3392
n3div36	1	> 7201	1	> 7201	1	> 7201	1	> 7201	1	> 7201	1	> 7202
n3seq24	4	> 7312	4	> 7222	5	> 7311	6	> 7246	1	! 2140	4	> 7213
n4-3	43434	1677	37108	1195	37831	1479	36245	1416	36261	1403	39036	1504
neos-1109824	8268	355	8164	355	8164	245	8164	360	8164	251	7810	370
neos-1337307	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
neos-1396125	38909	4258	35099	2834	28818	2812	31703	2699	34698	2802	33686	2792
neos13	1	> 7201	88	2085	63	1862	559	2093	284	2439	337	1880
neos-1601936	9458	6936	1	> 7200	4556	5120	3711	3677	2123	3326	4969	5463
neos18	5830	73	6318	77	6318	78	5703	69	5450	54	6318	77
neos-476283	372	2632	495	2473	397	2598	443	2631	445	2493	441	2580
neos-686190	3241	164	2692	138	2704	145	2751	144	2767	140	2705	144
neos-849702	20907	803	20907	863	20907	859	17371	856	17371	803	17371	859
neos-916792	66213	933	67052	931	57690	1618	65350	971	67925	947	63971	1063
neos-934278	45	> 7229	41	> 7214	9	> 7227	54	> 7226	1	> 7202	49	> 7215
net12	1	> 7201	992	> 7202	2001	6359	2412	6237	2119	6391	2418	6973
netdiversion	1	> 7371	1	> 7609	1	> 7562	1	> 7501	1	-	1	! 348
newdano	2058508	7024	2176391	5912	1976816	5849	1	> 7200	1659115	6580	1818678	5683
noswot	324848	260	345185	226	281390	185	362474	238	365152	196	362474	222
ns1208400	691	534	691	722	691	691	691	698	691	720	691	705
ns1688347	4538	602	2192	382	1449	311	1435	237	2185	357	2029	299
ns1758913	2	> 7881	-	-	-	-	-	-	-	-	-	-
ns1766074	934754	1871	934754	1833	934754	1884	934754	1916	934754	1896	934754	1893
ns1830653	60186	1178	48303	1034	33323	747	36032	820	37907	799	46111	882
opm2-z7-s2	1978	5101	1627	5261	1714	5706	1365	3799	1768	5599	1602	5762
pg5_34	253046	4173	259057	3129	250574	3126	290705	3606	260489	3148	304596	3885
pigeon-10	1	! 3071	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	! 3681
pw-myciel4	296556	4666	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
qiu	9811	91	9975	87	9692	87	9917	89	9713	87	9961	88
rail507	1	> 7202	1	> 7202	1	> 7205	1	> 7202	1	> 7204	1	> 7203
ran16x16	356748	499	418668	441	299623	346	302695	400	423285	452	316070	546
reblock67	83043	482	83902	489	83293	476	96807	439	81377	452	84623	529
rmatr100-p10	901	515	881	405	886	521	768	428	881	408	881	446
rmatr100-p5	397	2171	397	2121	403	2377	397	2279	397	2317	397	2098
rmine6	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
rocII-4-11	11288	443	10012	407	11029	418	12366	461	9107	486	10104	397
rococoC10-001000	1	! 4231	364404	3613	351486	3628	1	! 4297	596605	5416	633055	6006
roll3000	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
satellites1-25	11313	> 7203	3479	> 7201	1	> 7201	3479	> 7202	11745	6990	9260	> 7204
sp98ic	1	> 7201	1	> 7201	1	> 7201	1	> 7201	1	> 7201	1	> 7201
sp98ir	4999	223	4939	164	5555	173	4920	171	4416	161	5105	169
tanglegram1	22	> 7264	22	> 7264	1	! 324	22	> 7231	22	> 7241	1	> 7381
tanglegram2	3	20	3	19	3	19	3	20	3	14	3	19
timtab1	1014862	1027	464378	460	491225	429	496261	495	498709	457	493857	530
triptim1	9	3183	9	3690	9	3746	9	3678	9	3783	9	3760
unitcal_7	1	> 7201	1	> 7201	1	> 7201	1	> 7201	1	> 7201	1	> 7201
vpphard	2	> 7215	2	> 7232	1	> 7213	1	> 7217	2	> 7208	2	> 7208
zib54-UUE	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200	1	> 7200
solved/timeout/abort	56/27/4		56/28/2		58/25/3		56/27/3		59/23/3		57/24/5	
all solved: 52/87 geom. mean: 605.81	5128	706.1	4708	615.4	4440	594.9	4322	610.5	4551	588.8	4330	619.5
speedup: 1.166				1.15		1.19		1.16		1.20		1.14
diff. # of solved: +1.2				+0		+2		+0		+3		+1
geom. mean (comp.): 1551.98		1723.6		1557.3		1515.9		1551.8		1512.0		1622.8
all solved: 53/86 geom. mean: 617.52			4368	629.7	4097	607.8	4158	624.8	4319	604.8	4126	632.6

やはり安定的に多くのインスタンスを解くといえる．全ての試行で解かれたインスタンス数の平均という観点では，Normal Ramp-Up が 4 SOLVER スレッドから 8 SOLVER スレッドに増えるにつれ少なくなったのに対して，Racing Ramp-Up の方は増えている．どちらをデフォルト設定にするかの判断は難しいが，Racing Ramp-Up の方が安定的に解けている点と，並列化の規模が増えた際の効果が有利に働いていることから，Racing Ramp-Up をデフォルト設定としている．

最後に，本論文で行った数値実験とは実行環境，実験の条件，評価方法などの点において違いはあるが，FiberSCIP の並列 MIP ソルバとしての性能は H. D. Mittelmann により評価され WEB ページに公開されている (http://plato.asu.edu/bench.html, ここでは，ug [SCIP/*] ソルバとして示されている)．最新の並列 MIP ソルバとしての性能に関しては WEB ページを参照されたい．また，本論文においては，ページ数の制限もあるため MIP に対する結果のみを示した．FiberSCIP は，本原稿執筆時点でも，混合整数非線形計画問題 (MINLP: Mixed Integer

表 6　Summary of Speedups

Settings		Run-1	Run-2	Run-3	Run-4	Run-5	average
Only racing without comm.	Speedup						1.084
2 threads. all solved: 58(60)/87	diff. # of solved						+2.0
Only racing without comm.	Speedup						1.131
4 threads. all solved: 56(60)/87	diff. # of solved						+2.0
Only racing with comm.	Speedup	1.15	1.16	1.11	1.17	1.16	1.150
4 threads. all solved: 56(60)/87	diff. # of solved	+3	+4	+5	+4	+5	+4.2
Racing Ramp-Up	Speedup	1.29	1.24	1.23	1.27	1.27	1.261
4 threads. all solved: 53(62)/87	diff. # of solved	+6	+8	+6	+6	+6	+6.4
Racing Ramp-Up (no presolve)	Speedup	1.12	1.12	1.12	1.12	1.10	1.116
4 threads. all solved: 52(60)/87	diff. # of solved	+6	+8	+5	+6	+8	+6.6
Normal Ramp-Up	Speedup	1.70	1.65	1.67	1.67	1.65	1.668
4 threads. all solved: 52(59)/87	diff. # of solved	+6	+7	+7	+7	+6	+6.6
Normal Ramp-Up (no presolve)	Speedup	1.56	1.57	1.56	1.48	1.55	1.545
4 threads. all solved: 51(57)/87	diff. # of solved	+3	+3	+2	+4	+3	+3.0
Only racing without comm.	Speedup						1.008
6 threads. all solved: 55(59)/87	diff. # of solved						+1.0
Only racing without comm.	Speedup						0.854
8 threads. all solved: 50(56)/87	diff. # of solved						−2.0
Only racing with comm.	Speedup	1.00	1.01	0.99	1.05	0.97	1.005
8 threads. all solved: 49(53)/87	diff. # of solved	−2	+0	−2	+1	−1	−0.8
Racing Ramp-Up	Speedup	1.43	1.39	1.38	1.43	1.37	1.402
8 threads. all solved: 51(60)/87	diff. # of solved	+9	+7	+8	+7	+6	+7.4
Racing Ramp-Up (no presolve)	Speedup	1.15	1.15	1.10	1.12	1.16	1.135
8 threads. all solved: 50(58)/87	diff. # of solved	+5	+4	+5	+4	+6	+4.8
Normal Ramp-Up	Speedup	1.94	1.88	1.95	1.82	1.90	1.898
8 threads. all solved: 50(58)/87	diff. # of solved	+6	+4	+5	+4	+6	+5.0
Normal Ramp-Up (no presolve)	Speedup	1.95	1.86	1.90	1.85	1.89	1.891
8 threads. all solved: 48(57)/87	diff. # of solved	+3	+1	+4	+4	+3	+3.0

Nonlinear Programming) を扱うことができる．SCIP の MINLP ソルバとしての性能は，商用ソルバと比較しても遜色はないため，FiberSCIP はその性能をより強化することが期待されている．

4.2　ParaSCIP の数値実験結果

ParaSCIP による最初の成果は， スーパーコンピュータ HLRN II(https://www.hlrn.de/home/view) を最大 2,048 コア利用し, 最適解が得られていなかった MIPLIB2003(http://miplib.zib.de/miplib2003/index.php) の 2 つのインスタンス ds, stp3d に最適解を与えたことである．ds は，656 制約式，67,732 の 0-1 変数をもつ問題である．stp3d は，159,488 制約式，204,880 の 0-1 変数をもつ問題であるが，手動で 9 回前処理を行い，88,388 制約式，123,637 の 0-1 変数をもつ問題へ変換したものを解いている．いずれも大規模な問題ではあるが，特に stp3d は緩和問題も巨大で，その計算に時間を要する．計算結果は 10 回以上の再スタートにより求められた．詳細に関しては Shinano et al. (2012) を参照されたい．ここでは，Shinano et al. (2012) に書かれている内容は除き，その後に行った数値実験について述べる．論文としては，結果の新規性に欠けるかもしれないが，研究分野の実情を知る上

では有意義ではないかと考える．

まず，当時の状況として，いずれのインスタンスに対しても，ZIB 内では長時間の CPLEX の実行により，暫定解は保持しており，いずれの暫定解も最適解，あるいは，最適解にかなり近い解であると考えられていた．stp3d に関しては，暫定解が最適解であることの確認計算に終わったが，一方，ds に関しては，暫定解の目的関数値を 20% 近く更新した．そこで，CPLEX 12.0 に得られた解を与えて，CPLEX による計算を，ZIB が所有する 8 Quad-Core AMD Opteron 8384 プロセッサ， クロック 2.7 GHz を 4 個 (32 コア) 搭載し，512 GB メモリを持つ Sun Galaxy 4600 により行った．約一ヶ月程度の計算の後，CPLEX は異常停止し，その間，ParaSCIP により求まった解を更新することは無かった．そこで，再スタートを行わないで，1 ジョブにより HLRN II 上で ds インスタンスを ParaSCIP を用いて解くことを試みた．

HLRN II のジョブキューは，1 ジョブ当たり最大利用可能コア数は 2,048 で，最長実行時間は 12 時間が標準である．この標準仕様のジョブキューでは，再スタートを行わない実行により解くことは無理であると考えられたので，1 ジョブの上限利用可能コア数 4,096，最長実行時間 48 時間のジョブキューが用意され，解ける見込みがある場合には制限時間の手動による延長が可能となり，1 ジョブで解くことに必要なハードウェアが用意された．一方，ParaSCIP には，当時，Racing Ramp-Up の実装はなかった．Racing Ramp-Up は，1 ジョブによる計算を実現するために開発された．Racing Ramp-Up は，単純なアイデアの実装ではあるが，4,000 以上の異なるパラメタセットに対して安定して動作させるという実装上の困難さがあった．また，動的負荷分散に関しても，1 ジョブでの計算を実現するためには，収集モードとなる Solver 数を制限するとともに，下界値が上昇した場合には，より迅速に収集モードとなる Solver を切替えるなどの工夫を要した．さらに，実行時に 1 計算ノードでもメモリ容量を超えると，その計算ノードで動作していた Solver は停止し，1 Solver でも停止すると，MPI の仕様により全プロセスが停止するので，1 ジョブでの実行はメモリ使用量に注意する必要があった．しかし，本研究で使用メモリ容量を抑えるパラメタ設定をすることは，探索規則を早期に深さ優先へ切り替えることを意味し，これは無駄に多くの部分問題を解くことになる可能性が高くなる．よって，可能な限りメモリ容量制限まで利用することが好ましい．この種のトレードオフがあるため，パラメタ設定は容易ではない．

表 7　Single job computations

Instance name	cores	Time (sec.)	Solved Nodes	Transferred Nodes (% of Solved)	Average Idle Ratio (Max.)
ds	4,096	273,157	3,010,465,526	43,336,562 (1.4 %)	4.3 % (10.0 %)
stp3d (racing)	4,096	158,595	9,779,864	434,344 (4.4 %)	3.9 % (5.0 %)
stp3d (racing)	7,168	118,386	10,328,112	549,226 (5.3 %)	6.8 % (8.0 %)
stp3d (normal)	4,096	225,577	10,573,696	880,850 (8.3 %)	25.0 % (36,9 %)

表 7 に，HLRN II 上で実現した 1 ジョブでインスタンスを解いた際の実行結果を示す．stp3d に関しては，最適解を与えて計算を開始している．したがって，最適性の保証計算である．各インスタンスに対して，計算時間，解かれた部分問題数，転送された部分問題数，転送された部分問題数が解かれた部分問題数に占める割合，動作させた全ての Solver における遊休時間の割合の平均値および最大値が示されている．stp3d に関しては 4,096 コアを利用した Normal Ramp-Up の結果，および，7,168 コアを利用した Racing Ramp-Up の結果も示している．いずれの結果においても，Racing Ramp-Up 後の負荷分散が良好であることがわかる．ds では，解かれた部分問題数に対して 1.4% の部分問題の転送により負荷分散が実現しており，全 Solver の遊休時間の割合は平均 4.3%，最大でも 10% である．7,168 コアによる stp3d においても，解かれた部分問題の 5.3% にあたる部分問題の転送により負荷分散が実現しており，全 Solver の遊休時間の割合は平均 6.8%，最大でも 8% である．加えて，4,096 コア, 7,168 コアの両 Racing Ramp-Up の結果において解かれた部分問題数は，4,096 コアによる Normal Ramp-Up の結果において解かれた部分問題数より少ない．stp3d では，最適解を与えて計算を開始しているので，Normal Ramp-Up

の方が最初から並列に解く分，速く解けるという予測もできる．このような予測も，インスタンスに依存して変わる．stp3d の場合は，緩和問題の規模も大きいために，Normal Ramp-Up にも時間を要する．このことは，Normal Ramp-Up における遊休時間の割合の大きさからもわかる．遊休は，主に Ramp-Up 時に生じている．このような場合には，むしろ 1 Solver である程度の部分問題を生成してから分配する Racing Ramp-Up が有効となる．最適解を与えて開始した場合ですら，分枝変数の選択順序により解かれる部分問題の数は変わる．Racing Ramp-Up の勝者の生成した有望な分枝限定木を継続して解くメカニズムは，最適解を与えて計算を開始した場合においても有効であった．4,096 コアを 1.75 倍スケールアップした 7,168 コアにより stp3d の計算は 1.34 倍速くなっている．この規模のスケールアップとしては十分な加速が得られたと考えられる．

表 8　Checkpointing and restarted jobs computation

Run	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Acum. Time
Cores	512	1024	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	2048	
Time [h]	4	5	10	7	4	5	4	5	5	4	4	5	10	4	4	12	4	181,248

次に，チェックポイント・再スタートによる計算結果を，1 ジョブによる結果と比較する．Shinano et al. (2012) では ds インスタンスが 16 回再スタートによる 17 ジョブで解かれ，計算時間が約 86 時間と報告されている．この時間は，チェックポイントまでの時間を足し合わせて求められている．つまり，チェックポイント以降無駄にした時間は除外されている．本論文では，実際に利用した計算資源の総量を 1 コアで計算した場合の計算時間に変換して，1 ジョブでの計算結果と比較する．表 8 に，各ジョブと利用したコア数および実行時間 (Time[h]) を示し，1 コアに換算した総計算時間を示す．実際に，計算資源の利用時間は，チェックポイント後の無駄にした時間を含めると約 96 時間であり，1 コア利用の場合に換算すると全体で 181,248 時間になる．一方，4,096 コア利用して 1 ジョブで解いた結果を，1 コア利用の場合に換算すると，$4,096 \times 76 = 311,296$ 時間である．チェックポイント・再スタートでは，再スタート時に，前のジョブで行った計算の多くを破棄し，再スタート時には状態の回復に時間を要する．この回復の計算と，チェックポイント後に無駄にした計算を含めてさえ，チェックポイント・再スタートの方が少ない計算資源量で問題を解いたことになる．これは，1 ジョブでの実行が，早期に深さ優先探索に切り替わるメモリ不足に関して保守的な設定になっているのに対して，チェックポイント・再スタートを利用した場合には，前の結果に応じて実行時パラメタを調整しており，メモリ不足に関して保守的な設定をしていないことに起因する．実際，表 8 の実験では常に時間制限 12 時間のジョブとして実行していたが，各ジョブの計算時間は大きく異なっている．これは，多くの場合，メモリ不足による異常終了で計算が中断しているためである．極めて困難なインスタンスの場合には，最適解が得られやすい方向へ計算資源を集中させることが期待できるチェックポイント・再スタートの方が，1 ジョブでの実行と比較した場合，大量の計算を破棄しても有効に機能する場合があることがわかる．一方，大規模環境において 1 ジョブで問題を解くということに対するチャレンジによって，負荷分散方式を大幅に改善する結果に繋がったと確信している．大量の計算資源を必要とするので，それを確認するような計算実験の実施は現実的ではない．しかし，負荷分散がある程度有効に機能しているかどうかを確認するために，大規模環境において 1 ジョブで解くことを試みることは極めて有効である．分枝限定木全体を集中管理しない場合，1000 を超える Solver 数になると，多くの Solver が無駄な部分木の探索に多くの時間を費やすことを回避するためには実装上の細かな工夫が必要となる．そして，そのような工夫が有効に機能しないとオープンインスタンスを 1 ジョブで解くことは困難である．

MIPLIB2010 が公開され，最適解が得られていない多くのインスタンス (オープン・インスタンス) がライブラリに加わった．ParaSCIP は，それらに最適解を与えることが期待されており，Koch et al. (2011) が掲載される時点で，4 つのインスタンスには最適解を与えていた．その後，商用ソルバのバージョンが上がる毎に，多くのオープン・インスタンスが商用ソルバにより解かれている (http://miplib.zib.de/ の News に最新情報が示されている)．ParaSCIP は，統計数理研究所のスパコン Distributed-memory supercomputer Fujitsu PRIMERGY RX200S5 上

でも動作しており，LP ソルバとして CPLEX 12.3 を利用して，MIPLIB2010 の dg012142 (6,310 制約，640 の 0-1 変数，1,440 の連続変数を持つ MIP) を解いた．このインスタンスは，256 コアを利用して，約 42 時間の 1 ジョブによって解かれた．解かれた部分問題数は 941,693,415 で，最適値は 2,300,867 であり，MIPLIB2010 のページにも掲載されている．

5 おわりに

本論文では，現在も活発に開発が続けられている，制約整数計画ソルバ SCIP の並列化と，その性能を示す数値実験結果を紹介した．SCIP は商用のソルバと比較すると MIP ソルバとしての性能差はあるが，拡張性のある極めて質の高いソースコードが公開されている．したがって研究目的として利用する，あるいは，特別な用途のための専用ソルバを開発するという用途には適している．FiberSCIP は，SCIP が扱える全てのクラスの最適化問題を並列化できる状態になっており，MIP ソルバとしての性能は商用ソルバに劣っても，MINLP に対する並列ソルバとしての性能は，FiberSCIP が安定して動作する状態になればかなり期待できる．一方，ParaSCIP は，現在 MIP しか扱えない状態であるため，SCIP の扱える全てのクラスの最適化問題に対して動作することが期待されており，それを実現することは今後の課題である．また，並列ソルバ開発の効率化には，性能が著しく悪くても，決定性並列で動作する並列ソルバ，つまり，同じ動作環境，同じパラメタ設定に対しては，同じ分枝限定木を生成するソルバの開発が必要であり，これも今後の課題である．

最適化ソルバの性能を示す結果を見る際には，どのような評価により比較されているのかという点に十分注意する必要がある．本論文の実験結果で示したように，分枝限定法をベースとする並列ソルバの性能評価は，どのような評価用インスタンスセットと評価値を用いるかによって大きく変わる．制限時間内に解けるインスタンスを解く速度を重視するのか，制限時間内に解けるインスタンス数を重視するのか，あるいは，H. D. Mittelmann による評価のように総合的な評価指標を重視するのか，評価方法も様々であり確立されているとは言えない．FiberSCIP, ParaSCIP は，ベータ版ではあるが SCIP Optimization Suite に同胞され，ダウンロードして利用することが可能である．まだ，並列ソルバの研究としては道具が揃ってきた段階であるため，アルゴリズム改善等には多くの課題がある．例えば，並列動作する Solver 数の規模に応じて最も効率的に動作させるには，どのような機構が有効であるかということや，インスタンスとアーキテクチャの適性に関しても調べる必要がある．スーパーコンピュータで解くことを考えると，省電力で動作することも重要な課題である．しかし，現状では，再スタートによる実験結果が示すように，途中で多くの計算結果を破棄することが良い場合もあり，アルゴリズム改善を優先する必要があると思われる．加えて，ソルバ性能に対する評価方法も決定的なものはない．ベータ版ではあるものの，FiberSCIP と ParaSCIP が公開されたことを契機に，研究を加速することが期待されている．

謝辞

品野勇治と Thorsten Koch は，本研究に対して Google Research Grant のサポートを受けている．JST CREST : ポストペタスケールシステムにおける超大規模グラフ最適化基盤研究代表の藤澤克樹教授には，開発・デバッグ時に研究室の計算資源を一部使わせて頂いた．ご厚意に感謝致します．HLRN II の大量の計算資源を長期間利用させて下さり，かつ，HPC の技術サポートをしてくださった ZIB の HPC 部門のスタッフの皆様に感謝致します．統計数理研究所のスーパーコンピュータを利用させて下さった，統計数理研究所の伊藤聡教授に感謝致します．また，技術的なサポートをして下さった統計科学技術センターの蛭田智則様に感謝致します．本論文の原稿を読んでコメントを下さった兵庫県立大学の藤江哲也教授に感謝致します．

参 考 文 献

Achterberg, T. (2007). Constraint Integer Programming. Ph.D. thesis, Technische Universität Berlin.

Achterberg, T. (2009). SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, **1**(1), 1—41.

Achterberg, T. (2011). Comparing SCIP to CPLEX. INFORMS 2011 ANUAL MEETING.

Achterberg,T., Koch, T. and Martin, A. (2006). MIPLIB 2003. *Operations Research Letters*, **34**(4), 361—372.

Bixby, R. E., Boyd, E. A. and Indovina, R. R. (1992). MIPLIB: A test set of mixed integer programming problems, *SIAM News*, **25**, 16.

Bixby, R. E., Ceria, S., McZeal, C. and Savelsbergh, M. (1998). An updated mixed integer programming library: MIPLIB 3.0. *Optima*, **58**, 12—15.

Bixby, R. E., Cook, W., Cox, A. and Lee, E. K. (1999). Computational experience with parallel mixed integer programming in a distributed environment. *Annals of Operations Research*, **90**, 19—43.

Bussieck, M. R., Ferris, M. C. and Meeraus, A. (2009). Grid enabled optimization with GAMS.*INFORMS Journal on Computing*, **21**(3), 349-362.

Chen, Q. and Ferris, M. C. (1999). FATCOP: A fault tolerant condor- PVM mixed integer program solver. Technical report, Madison.

Eckstein, J. (1997). Distributed versus centralized storage and control for parallel branch and bound: Mixed integer programming on the CM-5. *Computational Optimization and Applications*, **7**(2), 199—220.

Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R. E., Danna, E., Gamrath, G., Gleixner, A. M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T.K., Salvagnin, D., Steffy, D. E. and Wolter, K. (2011). MIPLIB 2010. *Mathematical Programming Computation*, **3**, 103—163.

Linderoth, J. T. (1998). Topics in Parallel Integer Optimization. Ph.D. thesis, Georgia Institute of Technology.

MIPLIB2010. http://miplib.zib.de/

Mittelmann, H. D.. Benchmarks for Optimization Software. http://plato.asu.edu/bench.html

Phillips, C., Eckstein, J. and Hart, W. (2006). Massively parallel mixed-integer programming: Algorithms and applications. *Parallel Processing for Scientific Computing* (eds. M. Heroux, P. Raghavan and H. Simon), 323—340.

Ralphs, T. K. (2006). Parallel Combinatorial Optimization, chapter 3. Wiley, USA.

Ralphs, T. K., Ládanyi, L. and Saltzman, M. J. (2003). Parallel branch, cut and price for large-scale discrete optimization. *Mathematical Programmming*, (**98**), 253—280.

SCIP. http://scip.zib.de/

Shinano, Y. and Fujie, T. (2007). ParaLEX: A parallel extension for the CPLEX mixed integer optimizer. *LNCS 4757*, In 14th European PVM/MPI User's Group Meeting (EuroPVM/MPI2004), 97—106.

Shinano, Y., Fujie, T. and Kounoike, Y. (2003). Effectiveness of parallelizing the ILOG-CPLEX mixed integer optimizer in the PUBB2 framework. *LNCS 2790*, In International Conference on Parallel and Distributed Computing (Euro-Par2003), 451—460.

Shinano, Y., Achterberg, T. and Fujie, T. (2008). A Dynamic Load Balancing Mechanism for New ParaLEX, ICPADS 2008, 455—462.

Shinano, Y., Achterberg, T., Berthold, T., Heinz, S., Koch, T. (2012). ParaSCIP – a parallel extension of SCIP, *Competence in High Performance Computing 2010* (eds. Bischof, C., Hegering, H. G., Nagel, W. E., Wittum, G.), 135—148.

Xu, Y., Ralphs, T. K., Ladányi, L. and Saltzman, M. J. (2009). Computational Experience with a Software Framework for Parallel Integer Programming, *The INFORMS Journal on Computing*, **21**, 383—397.