Numerical Optimizer V22 新しい並列計算機能のご紹介

NTT データ数理システム 石橋保身

1. はじめに

近年，複数のプロセッサコアを搭載したマルチコアプロセッサが主流であるため，ハードウェア性能を引き出すための並列計算が重要です．また，クラウドコンピューティングサービスを利用することで誰でも手軽に計算環境を用意できるため，可搬性のあるソフトウェアであることも重要です．数理最適化パッケージ Numerical Optimizer においても高性能で可搬性を持った並列計算へのニーズが高いことから，V22 では並列計算機能の更新を行いました．具体的な更新内容は次の通りです．

1. マルチスレッドライブラリのリプレース
2. 分枝限定法の並列化手法の一新

本稿ではこれらの内容についてご説明します．

2. マルチスレッドライブラリのリプレース

Numerical Optimizer V21 以前はマルチスレッドライブラリに Intel[1] TBB を使用していましたが，これを C++11 標準ライブラリに置き換えました．これにより計算実行時にスレッド数を指定するだけで wcsp や分枝限定法の並列計算機能をご利用できます．また，Linux 環境やランタイムライセンスのみの環境でも並列計算機能をご利用いただけるようになります．

3. 分枝限定法の並列化手法の一新

並列計算をおこなうプログラムを設計する上で，ハードウェア環境を考慮しておくことは重要です．例えば私たちが日頃使用するデスクトップ PC のようにプロセッサがメモリを共有する環境（共有メモリ）の場合はマルチスレッドによる並列計算が可能です．この場合，スレッド間で簡単にデータを共有できますが，異なるスレッドが同時に同じアドレスに書き込みをしないような排他制御が必要です．一方，スパコンや PC クラスタのようにプロセッサがメモリを共有しない環境（分散メモリ）の場合はプロセス間通信を用いた並列計算が必要です．

V21 の分枝限定法の並列化手法は共有メモリ環境で実行することを念頭に置いた設計です．しかし，クラウドコンピューティングサービスを用いることで PC クラスタを容易に構築できるため，今後は分散メモリ環境で実行するニーズも増えると考えられます．そこで，並列計算の設計を見直して分散メモリ環境でも実行できるように修正をおこなっています．さらに，並列計算の手法自体を見直して最新の手法を取り入れます．V22 ではこの新しい並列化手法の一部をご提供します．次の節では一新された並列化手法の内容をご説明します．なお，紹介する手法は Zuse Institute Berlin ご所属の品野勇治博士の手法 [1] に基づいています．

---

[1] Intel はアメリカ合衆国およびその他の国における Intel Corporation の商標です

3.1. Racing

Numerical Optimizer は分枝限定法を基本的な枠組みとし，前処理や切除平面法，質の良い実行可能解を早く得るための手法（primal heuristics）などを用いて問題を解きます．これらの手法には多くのパラメータがあり，パラメータを変更するだけで問題を解けるかどうかが大きく変わるため，事前にパラメータをチューニングしておくことは非常に重要です．しかし，問題を解く前から最も良いパラメータを決定することは困難です．そこで，事前に用意した良さそうな複数のパラメータを並行して走らせて問題を解く手法が racing（または concurrent MIP）です（図 1）．各スレッドは同じ問題を解くため冗長な計算を多く含みますが，各スレッドで異なる探索がおこなわれているため実行可能解を見つけやすくなります．発見された実行可能解は他のスレッドにも共有されるため分枝限定法の枝刈りもされやすくなるというメリットもあります．

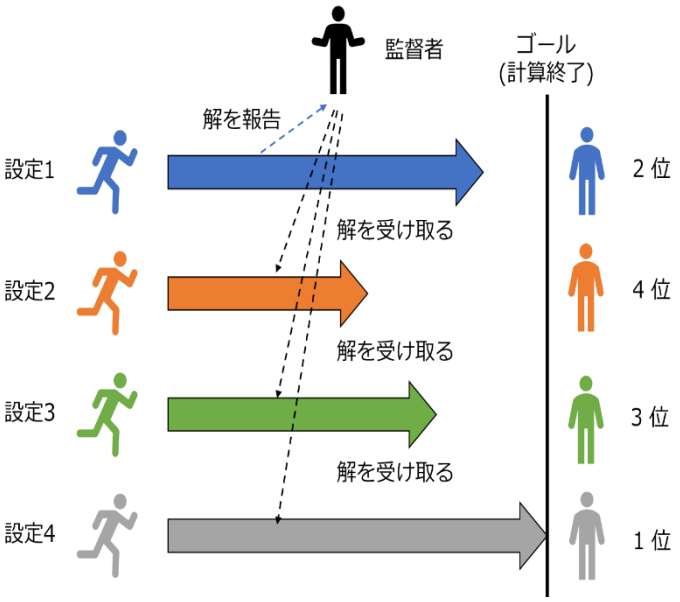

図 1　Racing

3.2. Deterministic Racing

並列計算は基本的に non-deterministic であるため，たとえ計算環境が同じであっても実行するたびに結果が異なります．例えば Racing の場合，得られる最適解は同じでも最も早く解き終わるスレッドが毎回異なるといったことが考えられます．これは他のソフトウェアと組み合わせて用いる場合に大きな障害となります．そこで，性能を犠牲にして Racing を deterministic にした Deterministic Racing を開発中です．Deterministic Racing で重要なことは解の報告と解の受け取り（つまりデータの共有）を常に同じタイミングでおこなうことです．これを達成するためにプログラムに一定間隔でバリアを設けて，バリアのタイミングで必ずデータを共有するようにします（図 2）．

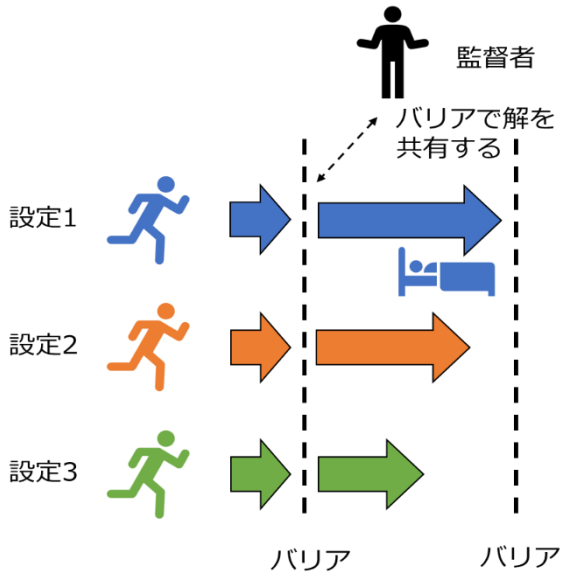

図 2　Deterministic Racing

4. 今後の展望

今回ご紹介した Racing の他にも探索木の部分木を並列に探索する Subtree の手法や，Racing と Subtree を組み合わせた手法も開発中です．これらの手法は Numerical Optimizer V23 でご提供予定です．

参考文献

[1] Y. Shinano, *et al*. “FiberSCIP A Shared Memory Parallelization of SCIP". In: *INFORMS Journal on Computing* 30.1 (2018), pp. 11-30.