A Massively Parallel Combinatorial Optimization Algorithm for the Costas Array Problem

Florian Richoux, Yves Caniou, Philippe Codognet, Reiji Suda

コスタス配置問題に対する超並列組み合わせ最適化解法

Florian Richoux
ナント大学ナント大西洋情報学研究所（LINA）
Yves Caniou
東京大学日仏情報学連携研究拠点（JFLI）
Philippe Codognet
東京大学日仏情報学連携研究拠点（JFLI）
須田礼仁
東京大学情報理工学系研究科

1. はじめに

並列計算は数値シミュレーションをはじめ，多様な分野で重要な技術となっている。近年では汎用プロセッサのマルチコア化が進み，モバイル端末も含めて複数のコアを有するCPUが主流になっている。また，従来はグラフィックス専用の付加プロセッサであったGPU（Graphics Processing Unit）が汎用化され，GPUの持つ多数の演算プロセッサが一般の計算にも使えるようになってきており，GPGPU（General Purpose computing on GPU）として定着している。今後も消費電力・発熱の観点から，CPUのクロック周波数の大幅な伸びは期待できず，半導体技術の進歩は並列性という形で情報処理速度に貢献することになる。このため，並列処理は，従来から広く研究されてきた数値計算などの分野を越え，今後はあらゆる情報処理技術の分野で用いられる基礎技術の一つなってゆかなければならない。

このような観点から，我々は離散組合せ問題を並列処理により効率的に解く研究を進めている。今回は大規模HPC Challenge の機会を得て，2014年8月28日にOakleaf FX10の全系を22時間30分用いた大規模計算に挑戦することができたので，これを中心に，我々の最近の研究成果について報告する。

2. 背景と関連研究

近年，局所探索（Local Search）系列のアルゴリズムとメタヒューリスティックスにより，実用的な大規模離散問題を成功裏に解くことができるようになっている。離散組合せ問題の多くは厳密に解くには指数関数的な計算量が必要と見られており，スーパーコンピュータをもってしても最適解を求めるのは困難である。このため，近似アルゴリズムが必須である。局所探索とメタヒューリスティックスに属するアルゴリズムとしては，Simulated Annealing，各種のGenetic Algorithm，Tabu Search，Swarm Optimization，Ant-Colony Optimization などがよく知られている。これらは乱数を用いており，Las Vegas アルゴリズムと呼ばれているアルゴリズムに属している。

離散組合せ問題の重要なクラスのひとつである制約充足問題（Constraint Satisfaction Problem: CSPと略されることもある）においても，古典的手法である制約伝搬型アルゴリズムでは到達できなかったような問題が，局所探索の手法により解くことができることが明らかになってきており，研究者の注目を集めている。制約充足問題とは，離散的な変数に対して加えられた一連の制約条件をすべて満たすような変数の値を求める問題である。制約充足問題とし

て古くから知られているものに魔方陣がある。魔方陣は，正方形n×nのますに数字を入れて，縦・横・斜めのいずれについても和が同じになるものである。すなわち，ますの数字が変数となり，縦・横・斜めのそれぞれについての和が制約条件となる。このほかｎクイーン問題などが古典的な制約充足問題としてよく知られている。

最近では計算機の主流は２コア，４コア，８コア，16コアといった並列計算機になってきており，制約充足問題などの離散アルゴリズムの研究分野においても，これらの計算機が提供する高い計算処理性能を活用し，効率的な並列実装を実現したいという期待が高まってきている。実際のところ，並列アルゴリズムの研究は制約充足問題の研究の黎明期からすでに始まっている。そのころの研究は，当時プログラミングのベースとされていた論理型プログラミング言語が持つ並列探索の機能を活用するものであった。

その後，様々な研究が行われ，並列実装が提案されてきたが，多くの場合は何らかの形でいわゆるOR並列性に基礎を置く実装であり，大規模な探索空間を小空間に分割し，それらを異なるコアに割り当てることにより並列化方式を採用していた。また，多くの研究では共有メモリ並列計算機を仮定して，大域的なメモリ空間に構築されたデータ構造を共有して，それを参照しながら各コアが部分計算を進めるというものであった。このような方式で効率的に並列実装ができるのは８コア程度であり，それより大規模な並列計算機を効率的に活用する制約充足問題のアルゴリズムは少数であった。

その後は徐々に並列処理の効率的な実装に関する研究が進んでいる。Comet systemは制約充足問題の解法の実装に関する一連の研究であるが，局所探索型アルゴリズムと制約伝搬型アルゴリズムの両方を並列実装し，小規模なPCクラスタで並列計算を実現したものがある[1,2]。最近になって，12コア程度まで並列実装が進んでいる。PaCCS[3]はさらに進んだ研究で，100コアのオーダーまで高い性能を達成することが報告されている。

また，同じ離散組合せ問題に属する問題である充足可能性問題（Satisfiability Problem: SATと略されることがある）は，制約充足問題を　{0, 1}　の２値に制約した有限値域制約充足問題とみなすことができるが，これに関しては研究が多く進められており，いくつかのマルチコア並列実装，さらには大規模なPCクラスタでの実装も行われている。

これに対し我々は，制約充足問題を数千コア以上の大規模並列プラットフォームにおいて効率的に並列化する研究に取り組んでいる。そのためには，従来手法で用いられてきた共有メモリ計算機を想定した共有データ構造の利用や，クラスタ型並列計算機で用いられてきた密に連携したマスター・ワーカー型並列制御ではなく，新たな形の並列性を検討する必要がある。我々の取るアプローチは，まず完全に独立な並列計算，もしくは限られた通信による並列計算で制約充足問題が効率的に解けるアルゴリズムを考える。現在，これが実現しつつある段階である。次のステップとして，これに何らかの通信を追加することによって実効性能を上げるの工夫ができるかを研究している。

スケーラブルな制約充足問題の並列解法を検討するにあたって，従来手法である制約伝搬型アルゴリズムをベースにして，大域的な制約グラフを多数のコアで共有して並列化するというアプローチでは，大規模な並列化が困難であることは明白と思われる。これとは別のアプローチとして，探索空間を分割して並列化する領域分割法などがある。これは可能性のある興味深い手法であるが，初期的な実装実験によれば，高性能化は数十コア程度で飽和する傾向にあることが知られている。例えば17クイーン問題の全解探索においては，32コアで28倍の高速化率

であるところ，64コアで29倍の高速化率しか得られなかったという報告がある[4]。最近ではより細粒度の領域分割法を用いることでより高いスケーラビリティが得られるという報告もある[5]。それによれば，制約充足問題の古典的なベンチマークにCSPLibがあるが，Gecodeをベースラインとしては40コアで平均14倍，or-toolsをベースラインとすると40コアで平均20倍の性能を達成している。しかし，唯一80コアで70倍の性能を得た17クイーン問題を除いて，40コアまでの性能しか報告されていない。

より広く組合せ最適化の研究において考えると，最初に大規模な並列計算が行われた手法は，古典的なアルゴリズムである分枝限定法であった。分枝限定法において効率的な大規模並列計算が成功した理由は，並列に走るプロセス間で通信しなければならない情報は，基本的には現在の上限値だけであり，通信の必要性が多くはなかったということが考えられる。このような背景から，グリッドコンピューティングにおいても最適化問題の解法として，分枝限定法が実装され評価されたという成果が知られている[6]。それによれば，数百コアに到るまで良好な高速化率が達成されたということであるが，興味深いことに，それよりもコア数が増大すると実行時間は一定になる傾向があると結論付けられている。また，プロジェクトスケジューリング問題に対する比較的シンプルな制約充足アルゴリズムを IBM Bluegene/P スーパーコンピュータに実装した事例[7]があるが，512 コアまではほとんど線形な高速化を達成したが，これを越えて1024コアまで実行したが高速化が達成されなかったということで，完全な成功とはいえない事例である。ごく最近になって，Limited Discrepancy Search という最適化アルゴリズムについて数千コアというレベルまで良好な高性能化を達成する並列化の結果がある[8]。この研究で重要な特徴として，並列に実行されるプロセス間でほとんど通信が必要でないということと，良好な負荷分散が得られるという 2 点が挙げられる。

3.我々のこれまでの研究

さらに異なるアプローチとして我々が採用しているのが，局所探索とメタヒューリスティックスによる手法である。ここでは，制約充足問題を最適化問題として定式化する。すなわち，充足していない制約の数を最小化するような目的関数を設定し，これを最適化のアルゴリズムであるメタヒューリスティックスを伴う局所探索によって解くことで，目的関数がゼロになったときに制約充足問題の解が得られるという仕組みである。局所探索に属するアルゴリズムの中で，特定の問題領域に依存しない汎用的な局所探索法として Philippe Codognet らにより提案されたアルゴリズムに，適応的探索法（Adaptive Search）がある[9,10]。適応的探索法では，ランダムな初期値からスタートして，満たされていない制約条件の数を減らすように，適当な目的関数を設定し，その目的関数を減少させるように，解を少しだけ変更する（近傍探索と呼ぶ）。このとき，目的関数は単純に満たされていない制約条件の数に設定するのではなく，うまく条件を満たす解に近づくように，問題ごとに工夫された適切な目的関数を選ぶことにより，効率的に問題を解くことができるのである。ただし，近傍探索だけでは一般の初期値から解には到達しない。このため，様々な初期値を取り，並列に探索する。この適応的探索法は，制約充足問題のような問題の構造を利用して探索を方向付ける局所探索のヒューリスティックスであり，線形の算術的制約条件，非線形の算術的制約条件，記号的制約条件など，広いクラスの制約に適用可能な手法である。また，適応的探索法は，本来的に過剰制約問題に適応する能力を持っている点でも利点がある。我々の提案手法は，この適応的探索法をベースとして，

独立な複数の初期値から局所探索をスタートさせることにより，極めて少ないプロセス間通信で並列化を可能にするものである。

我々の初期の実装[11]では，まず16 Cell/BE SPEコアを有する IBM BladeCenter で実験したところ，魔方陣，全音程を列挙する音列，完全正方形分割など，さまざまな古典的な制約充足問題に対してほとんど線形な高速化が得られた。さらに，このスケーラビリティが数百あるいは数千コアに到るまで維持されるような効率的な大規模並列実装を目指した。このために C 言語による逐次の適応的探索法の実装をもとにして， MPI を用いた並列プログラムを構築した。この実装は MPI が使える様々なプラットフォームで実行でき，PCクラスタ，スーパーコンピュータ，グリッドシステムでも動作する。性能評価においては，CSPLibにある制約充足問題のベンチマーク問題のほか，さらに難度の高い組合せ問題として，コスタス配列問題（Costas Array Problem）にも取り組んだ。このコスタス配列問題は，レーダーとソナーの周波数に関する通信分野の応用のある組合せ問題である。これらの問題を，東京大学のFX10とHA8000スーパーコンピュータ， Grid'5000 インフラストラクチャ（フランスの科学技術計算のための全国規模グリッドシステム），ならびにドイツのユーリヒスーパーコンピュータセンターにある JUGENEスーパーコンピュータにおいて実行してきている。これらのシステムは大規模並列計算機アーキテクチャの様々な方式を代表するものであり，我々の目標としては，特定のハードウェアや特定のアーキテクチャに依存した結果ではなく，できるだけ汎用的な知見を得ることを目指した。これまでにHA8000および Grid'5000 では 256 コア，JUGENEでは8192コアまで実行し，ほぼ線形な高速化率を達成してきている。今回の大規模 HPC Challenge ではFX10の76,800コアまで実行してきており，我々の知る限りにおいて，非自明な制約充足問題を数万コアで並列実行した事例は世界で最初となる。

3.研究の手法

今回の大規模 HPC Challenge は，2014年の 4 月から 8 月まで，著者の一人であるナント大学の Florian Richoux が日本学術振興会の短期滞在プログラムで東京大学に滞在した際に行われたものである。この滞在の目的は，これまでに著者らが進めてきた大規模並列計算機における組合せ最適化問題の新たなアルゴリズムのデザインと実装の研究をさらに推し進めるもので，特に東京大学の須田礼仁と Philippe Codognet と連携して，有限領域制約充足問題の高性能並列実装について研究を進めることである[12-16]。またその関連経費によりFX10のアカウントを得て，大規模な並列計算を実施することができた。その概要は下記に述べるが，日本とフランスの連携によって本研究を格段に進めることができた，有意義な滞在であった。

本研究において実装したのは，複数の制約充足問題アルゴリズムを実装したフレームワークであり，これを東京大学のスーパーコンピュータFX10に移植する作業から始めた。東京大学のFX10は76,800コアを有する大規模並列スーパーコンピュータであり，極めて強力であるが，アーキテクチャは独特のものがある。

このため，まず適応的探索法のプログラムを修正して，FX10で走るようにする必要があった。従来の並列バージョンの適応的探索法のプログラムは C 言語で MPI を用いて並列化されているため，大規模な改変は必要なく，これまでの日本，フランス，ドイツにおける複数のスーパーコンピュータの利用経験も役立ち，比較的短期間でポーティングを実現できた。

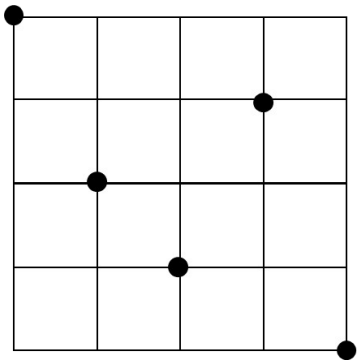
この実績に基づいて，東京大学情報理工学系研究科コンピュータ科学専攻の須田礼仁と，同

専攻に所属するJFLI（Japan-French Laboratory for Informatics: 日仏情報学連携研究拠点）の構成員であるPhilippe Codognet, Florian Richoux, Yves Caniou の連携で，東京大学情報基盤センターの大規模 HPC Challenge に応募した。この大規模HPC Challenge はFX10の全系が24時間利用できるというもので，我々の提案は採択されて，Florian Richoux の滞在期間のほぼ最後にあたる 8 月28日に割り当てられた。当日はキューの設定などに手間取り，実際に実行できた時間はおよそ22時間30分であったが，この間，FX10の全系76,800コアを最大限に活用するために，日本学術振興会の短期滞在のサポートによる 5 か月間の滞在を利用して，適応的探索法のFX10上での並列実装のために様々なテストやチューニングができたことに謝意を表したい。

4. 実装と結果

適応的探索法の並列実装のFX10へのポーティングは大きな問題なく実施でき，76,800コアの全系を用いて実行することができた。大規模 HPC Challenge で取り組む問題としては，適応的探索法が極めて良好に働くことが確認されている，コスタス配列問題を選んだ。

コスタス配列問題は，レーダーとソナーの周波数をノイズに強くなるように最適化する問題に由来するもので，割り当て問題のひとつである。サイズ n のコスタス配列問題は，n×nの格子を考え，そこに n 個の印をつける。その際，印は各行および各列には必ずひとつずつなければならず，また，2 つの印のすべてのペア（n (n-1)/2 通りある）を考えたときに，その格子上の相対位置が異なるベクトルとなるようにしなければならない。図 1 にサイズ 4 のコスタス配置問題の解の 1 つを示す。Vij は　i 番目の点と　j 番目の点の相対位置ベクトルである。



V12 = (1, 3)　　V24 = (2, -1)
V13 = (2, 1)　　V25 = (3, 1)
V14 = (3, 2)　　V34 = (1, 1)
V15 = (4, 4)　　V35 = (2, 3)
V23 = (1, -2)　　V45 = (1, 2)

図 1．コスタス配置問題

このように，コスタス配置問題は，記述すれば極めて単純であるが，2 つめの条件が計算を極めて複雑にする。コスタス配列問題の計算複雑性は指数関数的に増大し，ごく小規模の問題を除いて，厳密に解くことは極めて難しくなる。これまでにn=32のコスタス配列問題については，解があるかどうかも知られておらず，45年の間未解決問題として残っている。

そこで，我々はこのn=32のコスタス配列問題を大規模 HPC Challenge で取り組むこととした。これまでの実行から見られる傾向から，n=32の問題が解けるのは76,800コアで少なくとも24時間に近いオーダーと見積もられた。これはリスクのある選択であることはわかっていたが，成功すれば歴史的な成果となるため，我々はこのような選択をした。22時間30分の間　76,800コアを走らせて実行したが，残念ながらn=32の解は得られなかった。この結果は，下記のように，

適応的探索の実行時間の大きなばらつきから，ある程度予想された結果ではあった。さらに計算時間を延ばすと解が見つかる可能性もあるが，今回の結果から，そもそもn=32の問題に解がないという可能性も捨てられない。

まさしくチャレンジであったn=32の解は得ることができなかったが，滞在期間中にFX10を用いて大規模なコスタス配列問題を適応的探索法で解くことができた。例えば，n=22のコスタス配列問題を 512 コアと1152コアで解いたときに，スーパーリニアな高速化が達成された。また，我々が従来は達成できていなかった大規模な問題として， n=23，24，25 などの大きな問題も解くことができた。

表１．コスタス配列問題（n=22）の適応的探索法の並列所要時間

	512 コア	1152コア
平均値	256.80秒	85.67 秒
中央値	184.12秒	61.67 秒
最小値	0.66秒	1.41秒
最大値	938.85秒	189.26秒

表１は，n=22のコスタス配列問題を我々の並列適応的探索アルゴリズムを用いて， 512 コアと1152コアで25回ずつ実行したときの所要時間を示したものである。この実験では，平均値と中央値において，スーパーリニアな高速化が達成された。すなわち，コア数の比は2.25であるが，いずれもほぼ相対高速化率が 3 となった。なお，この例に見るように，適応的探索法の実行時間は大きくばらつきのあるもので，初期値に用いる乱数の値によって解が得られるまでの時間が大きく変わる。そのため，最大値では 512 コアの方が1152コアよりも時間がかかっているという事象も発生する。なお，我々の従来研究成果のひとつには，このように実行時間のばらつきのあるOR並列化において，所要時間の分布をモデル化し，並列所要時間を高精度にモデル化するものがある[17]。

表２．大規模なコスタス配列問題の適応的探索法の並列所要時間

問題サイズ	コア数	平均所要時間
n=23	1152コア	1,172.16秒
n=24	3456コア	37,886.15 秒
n=25	3456コア	1,190,288.46秒

表２には，大規模なコスタス配列問題を解いたときの平均実行時間を示す。すなわち，n=23からn=25までの大規模な問題を解かせた時の，平均の実行時間である。このように， n が大きくなるにつれてコスタス配列問題の複雑性は急激に高まっていくことがわかる。これからn=32の問題が45年の長きにわたって未解決問題として残されている理由の一端が示されている。

我々の適応的探索法の並列実装の最新版は，以下の sourceforge のリポジトリにおいて公開されている。

http://sourceforge.net/projects/adaptivesearch

5.適応的探索法の拡張

コスタス配置問題のほかに，我々は適応的探索法を拡張して，他の組合せ最適化問題に適用することも取り組んだ。今回の滞在時には，代表的な組み合わせ問題としてよく知られている巡回セールスマン問題を適応的探索法で解くことに取り組んだ。巡回セールスマン問題は，辺に重みの定義されたグラフに対して，グラフの頂点すべてを通る経路のうち，重みの和が最小となるものを求める問題である。特に今回は，グラフの頂点に 2 次元座標を割り当て，辺の重みを頂点間のユークリッド距離で定義する，ユークリッド的巡回セールスマン問題を取り上げた。これは巡回セールスマン問題の中でも基本的なもので，多くの関連研究があるため，比較が行いやすいという利点がある。また，須田研究室ではこの問題の大規模並列計算について研究成果があったことも理由である。

今回の適応的探索法の実装では，50点程度のグラフでは逐次計算で約10秒，12コアで約 1 秒で最適解を求めることができた。また， 100 点程度のグラフに対しては，最適解から0.5%以内の近似解を求めることができ，その所要時間は，逐次計算で約11秒，並列計算で 1 秒程度であった。さらに大きな問題として 130 点程度のグラフでは，最適解から0.5%以内の近似解を求めることができ，逐次計算で約 100 秒，並列計算で約10秒を要した。しかし， 150 点程度のグラフに対しては，現在の実装では効率的に解を求めることができなかった。今後は，大規模問題における効率の低下の原因を分析して，より効率的なアルゴリズムの開発を目指す。

また，適応的探索法を用いてゲームAI問題の解法の研究も行った。これまでの成果として，Build a wallという実時間戦略ゲーム（将棋や碁のようにプレイヤーが手を打つ順序が決まっているのではなく，実時間に進行する中で戦略を立てて手を打つ種類のゲーム）に対して効率のよい解法を与えることに成功している。今回， Florian Richoux の滞在期間中には日本のゲームAIの研究者と研究交流をして，実時間戦略ゲームにおける適応的探索法の可能性を模索した。この交流は今後の研究の進捗に重要な役割を果たすことが期待される。

なお，須田礼仁が担当し，情報基盤センターのFX10を教育利用で使用した大学院講義「並列数値計算論」においては Florian Richoux がゲストとして講演し，OR並列性を用いた離散アルゴリズムの並列化と，その精緻な性能モデリングについて紹介した。

5.まとめ

我々は，適応的探索法を並列化して，計算複雑性の高い組合せ問題に対して効率的な並列アルゴリズムを開発している。今回は大規模 HPC Challenge において東京大学のFX10全系76,800コアを用いて，45年来の未解決問題であるn=32のコスタス配置問題の解決に取り組んだ。残念ながらFX10全系を22時間30分用いても，n=32のコスタス配置問題の解を得るには至らなかったが，それに至るまでの今回の研究で適応的探索法の高い並列化効率が実証でき，また大規模な並列計算により大規模な問題を解くことができることを実証することができた。また，巡回セールスマン問題などの組合せ最適化問題に対して適応的探索法を用いて並列に解を求めることができることを示した。

今後は，適応的探索法の適用範囲を広めて，様々な離散組み合わせ問題に使えるアルゴリズムとして研究を進めてゆくとともに，並列実装においては通信を有効に用いてアルゴリズムに取り込み，近似解の探索の効率を高めることを目指す。

これから計算機のコア数はさらに増大し，マルチコアからメニーコアに主要アーキテクチャが移行し，ワークステーションでも数百コアを有するようになるであろう。このような時代に，様々な離散的な問題にも並列処理による高性能化が得られるように，我々の研究成果が活用されるように，今後も注力して研究を進めてゆくこととしている。離散最適化問題は応用範囲も広く，様々な現実問題において重要であるので，今後もこのような並列計算の研究は一層盛んになっていくものと思われる。

最後になりますが，大規模 HPC Challenge に取り組む機会を与えていただき，各種のサポートもいただきましたことに，情報基盤センターの各位に深い謝意を表します。本研究の一部は日本学術振興会 外国人特別研究員（欧米短期）の支援を受けています。

参 考 文 献

[1] L. Michel, A. See, P. Van Hentenryck, Distributed constraint-based local search, Proceedings of CP 2006, 12th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 4204, pp. 344-358, Springer, 2006.

[2] L. Michel, A. See, P. Van Hentenryck, Parallelizing constraint programs transparently, Proceedings of CP 2007, 13th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 4741, pp. 514-528, Springer, 2007.

[3] V. Pedro, Constraint Programming on Hierarchical Multiprocessor Systems, Ph.D Thesis, Universidade de Evora, 2012.

[4] L. Bordeaux, Y. Hamadi, H. Samulowitz, Experiments with massively parallel constraint solving. Proceedings of IJCAI 2009, 21st International Joint Conference on Artificial Intelligence, pp.443-448, 2009.

[5] J. C. Regin, M. Rezgui, A. Malapert, Embarrassingly parallel search. Proceedings of CP'2013, 19th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 8124, pp. 596-610, Springer, 2013.

[6] D. Caromel, A. di Costanzo, L. Baduel, S. Matsuoka, Grid'BnB: a parallel branch and bound framework for grids. Proceedings of HiPC'07, 14th International Conference on High Performance Computing, pp. 566-579, Springer, 2007.

[7] F. Xie, A. Davenport, Massively parallel constraint programming for supercomputers: challenges and initial results. Proceedings of CPAIOR'10, 7th International Conference on the Integration of AI and OR Techniques in Constraint

Programming for Combinatorial Optimization Problems, Lecture Notes in Computer Science, Vol. 6140, pp. 334-338, Springer, 2010.

[8] T. Moisan, J. Gaudreault, C. G. Quimper, Parallel discrepancy-based search. Proceedings of PC2013, 19th International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 8124, pp. 30-46, Springer, 2013.

[9] P. Codognet, D. Diaz, Yet another local search method for constraint solving. Proceedings of SAGA'01, International Symposium on Stochastic Algorithms: Foundations and Applications, pp. 73-90, Springer, 2001.

[10] P. Codognet, D. Diaz, An efficient library for solving CSP with local search. Proceedings of MIC'03, 5th International Conference on Metaheuristics, 2003.

[11] D. Diaz, S. Abreu, P. Codognet, Parallel constraint-based local search on the cell/BE multicore architecture. Intelligent Distributed Computing IV. Studies in Computational Intelligence, Vol. 315, pp. 265-274, Springer, 2010.

[12] Yves Caniou, Daniel Diaz, Florian Richoux, Philippe Codognet and Salvador Abreu, Performance Analysis of Parallel Constraint-based Local Search, 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2012), ACM Press, New Orleans (USA), February 2012.

[13] Daniel Diaz, Yves Caniou, Florian Richoux, Philippe Codognet and Salvador Abreu, Large-scale Parallelism for Constraint-based Local Search: The Costas Array Case Study, Constraints, 2014.

[14] Daniel Diaz, Florian Richoux, Yves Caniou, Philippe Codognet and Salvador Abreu, Parallel Local Search for the Costas Array Problem, Parallel Computing and Optimization (PCO 2012), IEEE, Shanghai (Chine), May 2012.

[15] Daniel Diaz, Florian Richoux, Philippe Codognet, Yves Caniou and Salvador Abreu, Constraint-based Local Search for the Costas Array Problem, Learning and Intelligent OptimizatioN Conference (LION 6), Springer LNCS, Paris (France), January 2012.

[16] Kamil Rocki, Martin Burtscher and Reiji Suda, The Future of Accelerator Programming: Abstraction, Performance or Can We Have Both?, XII International Conference on Parallel and Distributed Systems (ICPDS 2014), 442-443, Madrid

(Spain), March 2014.

[17] Charlotte Truchet, Florian Richoux, Philippe Codognet, Predicition of Parallel Speed-ups for Las Vegas Algorithms, Proceedings of ICPP 2013, 42nd International Conference on Parallel Processing, IEEE, 2013.