Rev. 1.1

シミュレーテッド分岐マシン TM

ユーザーマニュアル

Revision 1.1

© 2021 Toshiba Corporation / © 2021 Toshiba Digital Solutions Corporation

改定履歴表

版数	日付	変更内容
初版 ver.1.0	2021 年 3 月	新規作成
ver.1.1	2022 年 11 月	しおりを追加。誤字を修正

目次

第 1 章　はじめに....6
第 2 章　インストール....8
2.1　東芝への FPGA ボードの送付....8
2.2　動作環境セットアップ....8
2.2.1　ホストサーバー要件....8
2.2.2　ハードウェアのセットアップ....9
2.2.3　SBM ソフトウェアのセットアップ....15
2.3　SBM ソフトウェアによる回路 IP 演算実行....17
2.3.1　環境変数設定....17
2.3.2　診断プログラム....18
第 3 章　C ライブラリ....20
3.1　データ型....20
3.1.1　sbm_version_t....20
3.1.2　sbm_dtype_t....21
3.1.3　sbm_fpga_id_t....21
3.1.4　sbm_job_log_t....22
3.1.5　H_SBM_FPGA....23
3.1.6　H_SBM_JOB....23
3.2　API....23
3.2.1　sbm_setup_fpga....23
3.2.2　sbm_setup_fpga_id....24
3.2.3　sbm_create_job....25
3.2.4　sbm_set_data....26
3.2.5　sbm_set_initial_amplitude_x....28
3.2.6　sbm_set_initial_amplitude_y....29
3.2.7　sbm_run_sync....29
3.2.8　sbm_run_async....30
3.2.9　sbm_get_result....31
3.2.10　sbm_set_random_seed....31
3.2.11　sbm_release_job....32
3.2.12　sbm_close....32
3.2.13　sbm_set_message_level....33

3.2.14 sbm_print_lib_ver 33
3.2.15 sbm_get_lib_major_ver 34
3.2.16 sbm_get_lib_minor_ver 34
3.3 エラーメッセージ 35
第 4 章 Python ライブラリ 37
4.1 API 37
4.1.1 setup_fpga 37
4.1.2 setup_fpga_id 38
4.1.3 create_job 39
4.1.4 set_data 41
4.1.5 set_initial_amplitude_x 43
4.1.6 set_initial_amplitude_y 44
4.1.7 run_sync 44
4.1.8 run_async 45
4.1.9 get_result 46
4.1.10 set_random_seed 47
4.1.11 release_job 47
4.1.12 close 48
4.1.13 set_message_level 48
4.1.14 print_lib_ver 49
4.1.15 get_lib_major_ver 49
4.1.16 get_lib_minor_ver 50
4.2 エラーメッセージ 50
第 5 章 回路インスタンス一覧 51
第 6 章 API の使用例 52
6.1 同期型のサンプルコード［C］ 52
6.2 非同期型のサンプルコード［C］ 54
6.3 同期型のサンプルコード［Python］ 57
6.4 非同期型のサンプルコード［Python］ 58
第 7 章 リファレンスデザインチュートリアル 61
7.1 はじめに 61
7.2 マルチオブジェクトトラッキング 64
7.2.1 概要 64
7.2.2 二部グラフ最大マッチング問題 64
7.2.3 ファイル構成 65

7.2.4 実行環境の準備 66
7.2.5 実行方法 67
7.2.6 解説 68
7.3 インタラクティブ巡回セールスマン問題ソルバ 77
7.3.1 概要 77
7.3.2 ファイル構成 78
7.3.3 実行環境の準備 (CLI 版・GUI 版共通) 78
7.3.4 実行環境の準備 (GUI 版向け) 79
7.3.5 実行方法 80
7.3.6 解説 82
7.4 最大独立集合問題ソルバ 86
7.4.1 概要 86
7.4.2 Market Graph 86
7.4.3 ファイル構成 87
7.4.4 実行環境の準備 (CLI 版・GUI 版共通) 88
7.4.5 実行環境の準備 (GUI 版) 88
7.4.6 実行方法 88
7.4.7 解説 91
第 8 章 Appendix 97
8.1 Disclaimer 97
8.2 License 98
8.3 診断プログラムについて 101

第1章　はじめに

シミュレーテッド分岐マシン TM（SBM）は東芝が開発した組合せ最適化問題を高速に解くソルバです。本書において東芝は 株式会社 東芝 及び 株式会社 東芝デジタルソリューションズ を含む総称を意味します。本書および本商品において SBM は シミュレーテッド分岐マシン TM または Simulated Bifurcation Machine TM を意味します。

SBM は下記論文の理論に基づいて開発されています。

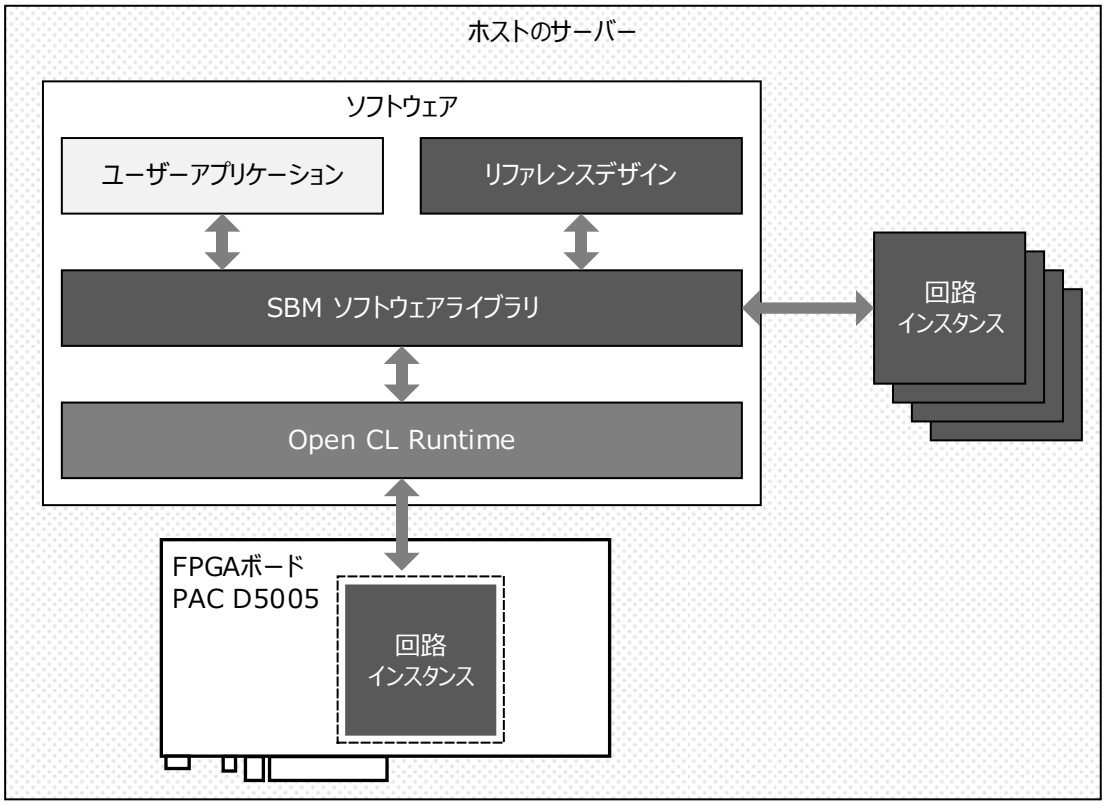
Goto, H., Tatsumura, K., & Dixon, A. R. (2019) Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems, *Science Advances*, 5(4), DOI:10.1126/sciadv.aav2372

Goto, H., Endo, K., Suzuki, M., Sakai, Y., Kanao, T., Hamakawa, Y., Hidaka, R., Yamasaki, M., & Tatsumura, K. (2021) High-performance combinatorial optimization based on classical mechanics, *Science Advances*, 7(6), DOI:10.1126/sciadv.abe7953

SBM は FPGA（field-programmable gate array）に実装されます。FPGA の回路書き換え可能という特長を活かして、プロファイルの異なる複数の SBM 回路（回路インスタンス）を必要に応じて動的に切り替えて利用できます。SBM は Intel FPGA SDK for OpenCL ソフトウェア・テクノロジー開発環境によって開発されており、Intel FPGA PAC D5005（Intel FPGA プログラマブル・アクセラレーション・カード D5005）専用の回路インスタンスとして提供されます。本書において FPGA ボードは、上記 Intel FPGA PAC D5005 を意味します。

SBM を応用したシステムは、図 1 に示すようにホストのサーバーとそこに装着された FPGA ボードから構成されます。ユーザーは、C、Python によるソフトウェアライブラリによって、ハードウェア処理を意識することなく、アプリケーションを開発できます。アプリケーション開発例としてリファレンスデザインが付属されます。以降、本書においてホストサーバーは上記ホストのサーバーを意味します。

図 1: SBM を応用したシステムの概略図



本システムの提供物は以下の通りです。

- SBM 回路 IP（Intellectual Property *1）（Intel FPGA PAC D5005 専用）
- SBM ソフトウェアライブラリ、API（C、Python）
- リファレンスデザイン
- ユーザーマニュアル（本書）

*1: 本書では、複数の SBM 回路インスタンスを総称して SBM 回路 IP と呼びます。

SBM を使用する際には、本マニュアルのライセンス条項「8.2 License」（P.98）を必ずお読みください。本商品には、特許出願済みの技術を使用しております。本書は当社の著作物であり、その全部あるいは一部について、当社の承諾を得ずに複製、改変若しくは第三者に提供することは法律で禁止されております。その他、本書に掲載の会社名もしくは商品名等は、それぞれ各社が商標として使用している場合があります。シミュレーテッド分岐マシンと Simulated Bifurcation Machine は、株式会社東芝の日本またはその他の国における登録商標または商標です。

第2章　インストール

2.1　東芝への FPGA ボードの送付

SBM 回路 IP を使用するには専用のインタフェース回路をあらかじめ FPGA ボードの不揮発メモリに書き込む必要があります。所有する FPGA ボードを送付頂き、東芝で専用のインタフェース回路を書き込んだ後、返送します。送付先については東芝担当者にご相談ください。

2.2　動作環境セットアップ

本節では、ハードウェアのセットアップおよびソフトウェアのセットアップを行います。

ハードウェアのセットアップ

CentOS 7 稼働サーバー上で、ハードウェアのセットアップを行います。FPGA ボード装着と、FPGA ボードが動作可能となるまでの Intel 製 FPGA 制御ドライバインストールを行います。

SBM ソフトウェアのセットアップ

東芝製 SBM 回路 IP および東芝製 SBM ソフトウェアを展開・配置して、回路 IP が演算実行可能となるまでの設定を行います。

次小節以降で、それぞれ順を追って説明します。

2.2.1　ホストサーバー要件

ホストサーバーは、FPGA ボード（Intel FPGA プログラマブル・アクセラレーション・カード D5005）を搭載可能で、Intel アクセラレーション・スタック バージョン 2.0.1 ランタイムを安定して実行可能で、セットアップ完了後には **OpenCL 実行環境確認コマンド「aocl diagnose all」が PASS すること**を要件とします。

参考情報として、Intel PAC 認定サーバーと、サーバー構成例を以下に示します（冷却性能等も重要であり、下記構成で必ず動作することを保証する意図ではありません）。

Intel PAC 認定サーバー[1]

HPE ProLiant DL380 Gen10、Dell PowerEdge R740、等

サーバー構成例

Intel CPU 8XXX 以上、DRAM 8GB 以上、PCIe-Slot Gen3 X16

2.2.2 ハードウェアのセットアップ

ハードウェアのセットアップを以下の手順で実施します。

1. ホストサーバーへの OS インストール
2. Intel 製 FPGA 制御ドライバをダウンロード
3. Intel 製 FPGA 制御ドライバをインストール

ホストサーバーへの OS インストール

ホストサーバーに対して、指定の OS をインストールしてください。SBM ソフトウェアは、Red Hat Enterprise Linux 互換 OS である CentOS 7.9 にて動作実績があります。

（ダウンロードサイトご参考）

```
http://isoredirect.centos.org/centos/7/isos/x86_64/
```

Intel 製 FPGA 制御ドライバのダウンロード

Intel アクセラレーション・スタック バージョン 2.0.1 ランタイムを下記 URL より取得します。「ランタイム用アクセラレーション・スタック」（約 350MB）を選択して取得します。

```
https://www.intel.co.jp/content/www/jp/ja/programmable/products/boards_and_kits/dev-kits/altera/intel-fpga-pac-d5005/getting-started.html
```

[1] https://www.intel.com/content/www/us/en/programmable/products/boards_and_kits/dev-kits/altera/intel-fpga-pac-d5005/buy.html

Intel 製 FPGA 制御ドライバのインストール

「ランタイム用アクセラレーション・スタック（RTE）」をインストールします。下記何れかのクイックスタートガイド（以降、Intel 社クイックスタートガイド）に従い、root アカウントで RTE をインストールします。

https://www.intel.com/content/www/us/en/programmable/documentation/edj1542148561811.html

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug-qs-ias-d5005.pdf

Intel 社クイックスタートガイドに従い、FPGA ボードの搭載および RTE のインストールを行います。以下、Intel 社クイックスタートガイドの各ステップで、**SBM ソフトウェアを動作させるための**インストール手順の補足を説明します。本稿での参照した Intel 社クイックスタートガイドは、UG-20202 | 2020.09.09 であり、以下の説明文中の見出し参照番号も同版に準じます。以下では SBM ソフトウェアを実行する上で必要となる最小限の手順として説明しますので、“実行不要”となっている手順等も用途に応じて実施して構いません。

パッケージ／ライブラリの事前インストール

必要となるパッケージ/ライブラリを一通り入れておくと、以降のインストール処理がスムーズに通ります（sudo コマンドでエラーが出る場合は、root アカウントで直接実行ください）。

```
$ sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
$ sudo yum install gcc gcc-c++ cmake make autoconf automake libxml2 libxml2-devel json-c-devel boost ncurses ncurses-devel ncurses-libs boost-devel libuuid libuuid-devel python2-jsonschema doxygen hwloc-devel libpng12 rsync python2-pip tbb-devel
$ sudo pip install intelhex
$ sudo yum install -y python36 python36-pip
```

> 2.3. Hardware Installation

12V 電源ケーブルの 2x4-pin PCIe PowerConnector への接続が必要です（PCIe スロットからの電力供給は不十分のため）。また、ホストサーバーの冷却性能として、下記資料 Table 12. Thermal Specifications で規定される風量を確保できていることをご確認ください。

```
https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ds/ds
-pac-d5005.pdf
```

> 2.4.1. Installing Red Hat Enterprise Linux version 7.6

SBM ソフトウェアは、Red Hat Enterprise Linux 互換 OS である CentOS の ver 7.9 で動作実績があります。Red Hat Enterprise Linux の場合は下記コマンドを実行してください（CentOS の場合は不要です）。

```
$ subscription-manager attach -auto
$ subscription-manager repos --enable rhel-7-server-rpms rhel-7-
server-optional-rpms
$ sudo yum groupinstall <package_name>
$ sudo yum install -y python36 python36-pip
```

> 2.4.2. Installing the Intel Acceleration Stack

Acceleration Stack for Runtime をインストールします。

> 2.4.2.1. Installing the Intel Acceleration Stack Runtime Package on the Host Machine

root アカウントで、以下コマンドを実行します。

```
 # ./setup.sh
```

インストール先ディレクトリは、

```
/home/root/intelrtestack/
```

ではなく、

```
/usr/local/intelFPGA_pro/intelrtestack/
```

等を推奨します。

4. You are prompted with the following questions:

上記指示では、以下、Yes を選択して全てインストールしてください。

```
 a.Do you wish to continue to install the software(Acceleration
Stack)?
 b.Do you wish to install the OPAE?
 c.Do you wish to install the OPAE PACSign package?
```

（重要）インストール過程で「パッケージが足りないためインストールできませんでした」というメッセージが出た箇所は、該当するパッケージを yum で入れなおしてから再度インストールを実施してください。

8. Run the initialization script to set the required environment variables and OPAE_PLATFORM_ROOT:

上記指示は、**今後回路 IP を動作させる際に、ログイン後もしくは新たな端末を開いた後、必ず実行する必要がある環境設定です。**

```
$ source /usr/local/intelFPGA_pro/intelrtestack/init_env.sh
```

> 2.4.2.2. Installing the Intel Acceleration Stack Development Package on the Host Machine

実行不要です。

> 2.4.3. Verifying the Installation

実行し、記載されているログが出力されることを確認してください。本マニュアルに従っても opae-1.1.4-8.x86_64 はインストールされませんが、問題ありません。

> 3. Installing the OPAE Software Package

実行不要です。

> 4. Identify the FPGA Interface Manager (FIM) and BMC Firmware Version

実行の上、下記 ID が出力されることを確認して下さい（2021 年 3 月版東芝製回路 IP 導入済 FPGA ボードの場合）。

```
Board Management Controller, microcontroller FW version 2.0.12, RTL
version 2.0.6
Bitstream Id                : 0x202000200000701
Bitstream Version           : 2.0.2
Pr Interface Id             : fbfdaf19-d278-5850-b97d-085fe209c44d
```

> 4.1. Updating the FIM and BMC

実行不要です。

> 5. Running FPGA Diagnostics

実行不要です。代わりに、OpenCL 実行環境組み込みを行うために、root アカウントで以下を実施ください。

```
# reboot
# source /usr/local/intelFPGA_pro/intelrtestack/init_env.sh
# aocl install
```

> 6. Running the OPAE in a Non-Virtualized Environment

実行不要です。

> 7. Running the OPAE in a Virtualized Environment

実行不要です。

> A. Handling Graceful Thermal Shutdown

必要に応じて、実行ください。

SBM ソフトウェアは回路実装密度が高く、FPGA 温度上昇に対するケアが重要です。SBM ソフトウェアを安定して動作させるため本 Thermal Shutdown 設定を推奨します。

Intel PAC 認定サーバーで冷却 FAN の風量を最大設定にしている場合、以下の設定は必ずしも必要ではありません。ホストサーバーで冷却 FAN の風量を意図的に弱く設定している場合や Intel PAC 認定サーバー以外で動作させる場合は、下記資料 Table 12. Thermal Specifications で規定される冷却能力を確保できる冷却 FAN の設置を推奨します。

https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ds/ds-pac-d5005.pdf

FPGA の junction 温度は fpgainfo temp コマンド（「FPGA Core Temperature」出力値）で確認できます。下記例を設定しておくと、FPGA の junction 温度が 95℃を超えた場合、Thermal Shutdown がかかります。設定の一例を以下に示します（/usr/bin/pacd.conf 記述内容、下線部はインストール場所に合わせてください）。下線部の「.gbs ファイル」は、次節ソフトウェアのセットアップにて SBM ソフトウェアの展開・配置を行った後に出現します。

```
############## REQUIRED OPTIONS ################
DefaultGBSOptions=-n /home/<username>/<user work dir>/sbm_v1_0_0/sbm/images/COOLDOWN.gbs
StartedBySystemd=1

############## OPTIONAL OPTIONS ################
UMask=0
LogFile=/tmp/pacd.log
PollInterval=10
CooldownInterval=10

# Uncomment and specify specific PAC PCI address to monitor.
```

```
#BoardPCIAddr=-S 0 -B 5 -D 0 -F 0

# Specify threshold values. -T for UNR, -t for LNR.
ThresholdOptions=-T 12:95.0:90.0

# Extra advanced options.
ExtraOptions=
```

> B. FPGA Device Access Permission

必要に応じて、実行ください。

> C. Memlock Limit

必要に応じて、実行ください。

> D. Troubleshooting Frequently Asked Questions (FAQ)

必要に応じて、参照ください。

2.2.3　SBM ソフトウェアのセットアップ

SBM ソフトウェアのセットアップを、以下の手順で実施します。

1. SBM ソフトウェア一式を入手
2. SBM ソフトウェアの整合性を確認
3. SBM ソフトウェアの展開・配置
4. SBM ソフトウェアの動作に必要なツール・ライブラリの導入

SBM ソフトウェア一式を入手

東芝製 SBM ソフトウェア一式は、DVD-R メディアにて送付されます。東芝担当者からの案内に従い同 DVD-R メディアを入手ください。

SBM ソフトウェアの整合性を確認

md5sum コマンドにより、下表のチェックサム値と同一であることを確認します。

```
$ md5sum sbm_common_derivative_v1_0_0.tgz
```

表 1: derivative checksum (md5sum value)

derivative (v1_0_0)	md5sum value
sbm_common_derivative_v1_0_0.tgz	171ce27d2dfc1b1d27c9c9a944c2af56

SBM ソフトウェアの展開・配置

tar コマンドにより、作業ディレクトリ下へ入手済ファイルを展開・配置します。

```
$ cp sbm_common_derivative_v1_0_0.tgz /home/<username>/<user_work_d
ir>
$ tar -xvzf sbm_common_derivative_v1_0_0.tgz
```

SBM ソフトウェアの動作に必要なツール・ライブラリの導入

Google Chrome、Java 用のライブラリ等を、以下のようにしてインストールします。別章にて説明する SBM リファレンスデザイン（デモ動作）の実行時には、個々のリファレンスデザイン実行手順に従って追加ライブラリの導入が必要になる場合があります。

Google Chrome をインストール

下記の何れかのコマンドでインストールしてください。

```
$ sudo yum install google-chrome-stable
```

```
$ sudo yum install https://dl.google.com/linux/direct/google-
chrome-stable_current_x86_64.rpm
```

Python GUI 系ライブラリをインストール

```
$ sudo yum install python3-tkinter
```

2.3　SBM ソフトウェアによる回路 IP 演算実行

本節では、SBM ソフトウェアによる回路 IP 演算実行の手順を示します。

1. SBM ソフトウェア実行用環境変数設定
2. SBM 回路 IP 診断プログラム

次小節以降で、それぞれ順を追って説明します。

2.3.1　環境変数設定

必要な環境変数（Intel 側環境）を設定します。参照先は Intel Acceleration Stack Runtime のインストール場所に合わせて、適宜読み替えてください。

```
$ source /usr/local/intelFPGA_pro/intelrtestack/init_env.sh
```

本スクリプトを一般ユーザーとして実行した場合、[sudo]コマンドに対するパスワード入力を求められます（root ユーザーとして実行した場合は、求められません）。

必要な環境変数（東芝側環境）を設定します。

```
$ cd /home/<username>/<user_work_dir>/sbm_v1_0_0
$ source ./setup.sh
```

（参考）setup.sh の中身：絶対パス表現

```
export SBM_HOME=$(pwd)
export SBM_IMAGES_DIR=${SBM_HOME}/sbm/images
export LD_LIBRARY_PATH=${SBM_HOME}/sbm/lib:$LD_LIBRARY_PATH
export SBMLIB_PATH=${SBM_HOME}/sbm
export PYTHONPATH=${SBM_HOME}/sbm/lib:$PYTHONPATH

aocl program acl0 ${SBM_IMAGES_DIR}/COOLDOWN.aocx
```

正しく設定が行われたかを確認します。

OpenCL 実行環境として正しく動作可能な状態であることを確認するために、下記コマンドを実行し、応答結果が “DIAGNOSTIC_PASSED” であることを確認してください。

```
# reboot
# source /usr/local/intelFPGA_pro/intelrtestack/init_env.sh
# aocl diagnose all
```

2.3.2 診断プログラム

診断プログラム（sbm_diagnose.sh）を使用することにより、SBM ソフトウェアの動作を診断することができます。診断プログラムを使う場合は以下の手続きを実施してください。

G-set 問題を定義するデータファイル一式を、下記 URL から取得して使用します。

```
http://web.stanford.edu/~yyye/yyye/Gset/
```

取得したデータを下記コマンド例に従って適切なディレクトリに展開してください。

```
$ cd ${SBM_HOME}/examples/data
$ cd G-set/
$ wget -r -l1 -np "http://web.stanford.edu/~yyye/yyye/Gset" -P ./ -
A "G*"
$ mv ./web.stanford.edu/~yyye/yyye/Gset ./Gset
$ rm -rf ./web.stanford.edu/
$ for filename in ./Gset/*; do mv $filename ${filename}.txt; done
```

TSP 問題を定義するデータファイル一式を、下記 URL から取得して使用します。

```
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/dantzig42.tsp.gz
```

取得したデータを下記コマンド例に従って適切なディレクトリに展開してください。 python 環境の構築は、ご使用状況により適宜変更ください。

```
$ cd ${SBM_HOME}/examples/tsp
```

```
$ wget http://comopt.ifi.uni-
heidelberg.de/software/TSPLIB95/tsp/dantzig42.tsp.gz
$ gzip -d ./dantzig42.tsp.gz
$ mkdir ../gui_server/data
$ cp dantzig42.tsp ../gui_server/data/
$ cd ../gui_server/
$ sudo yum install -y https://repo.ius.io/ius-release-el7.rpm
$ sudo yum install python36-devel
$ pip3 install -r requirements.txt --user
$ cd ../tsp/
$ python3 format_dantzig42.py
```

下記スクリプトにより、SBM 回路 IP を診断します。スクリプト実行結果例は「8.3 診断プログラムについて」（P.101）を参照ください。実行時間は約 10 分です。

```
$ cd ${SBM_HOME}/examples/problems_for_diagnose
$ ./sbm_diagnose.sh
```

SBM 回路 IP 診断実行時には、処理 CPU コアの切り替わりによる計時誤差（遅延時間増加）を極力排除すべく、搭載 CPU コアの一部を Linux Kernel（OS）処理から除外指定してから回路 IP 診断専用に同 CPU コアを割り当てて実行することを推奨します。32 コア CPU 搭載システムでの指定方法一例を以下に示します。

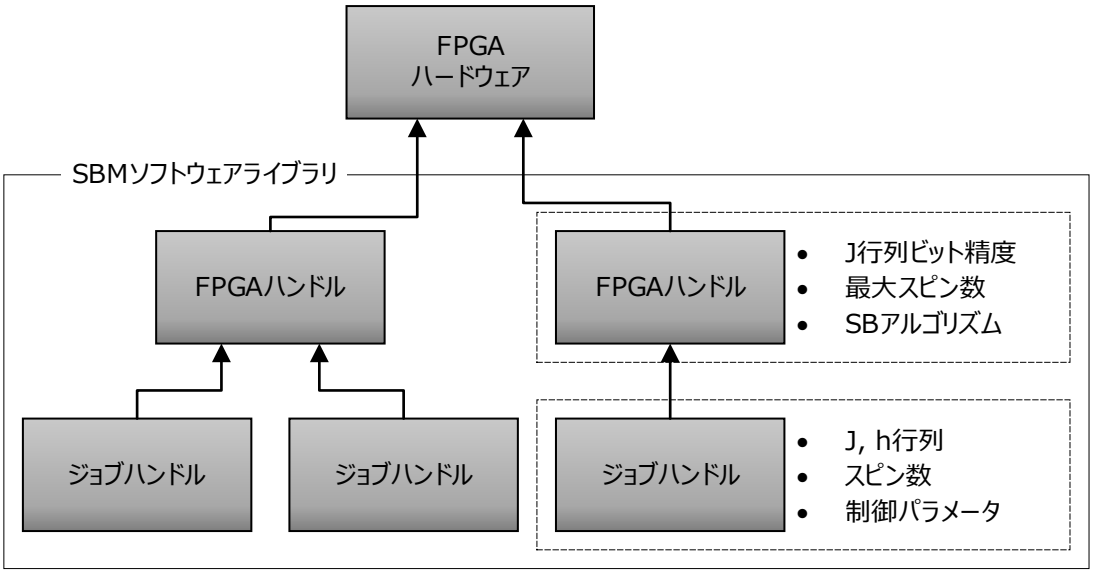
```
/etc/default/grub 内の、GRUB_CMDLINE_LINUX 指定行に、"isolcpus=30-31"を追記します（32 コア CPU の 30/31 番目は、Linux Kernel 処理をさせない指定）
$ sudo grub2-mkconfig -o /etc/grub2-efi.cfg
   （対象ファイル名は grub2.cfg という場合もあります）
$ sudo reboot

$ ps -eo pid | grep -v PID | xargs -n1 taskset -c -p | grep ': 0-
31' | sed -r -e 's/pid （[0-9]*）.*$/¥1/' | xargs -n1 sudo taskset -
ap 3fffffff
   （稼働中プロセスのうち、移動可能なものは CPU の 30/31 番目コアから退去させます）
$ cd ${SBM_HOME}/examples/problems_for_diagnose
$ taskset -c 30-31 ./sbm_diagnose.sh
   （CPU の 30/31 番目コア上で診断プログラムを実行します）
```

第3章　C ライブラリ

SBM ソフトウェアライブラリは、FPGA ハンドルとジョブハンドルの概念で構築されます（図 2）。FPGA ハンドルは回路インスタンスを抽象化したもので、サポートする回路インスタンスのうちのいずれか 1 つを指し示します。ジョブハンドルはジョブを抽象化したものです。実行すべきイジング問題の J 行列、h 行列および制御パラメータとそれらを動作させる回路インスタンス情報（FPGA ハンドル）を 1 つにまとめた単位をジョブと呼びます。複数のジョブハンドルが同一の FPGA ハンドルを参照することが可能です。

図 2: SBM ソフトウェアライブラリの概念図



3.1　データ型

3.1.1　sbm_version_t

SB アルゴリズムバージョンのリストです。

パラメータ

名称	説明
E_SBM_ALGO_A	SB バージョン A（*1）
E_SBM_ALGO_B	SB バージョン B（*2）
E_SBM_ALGO_C	SB バージョン C（*3）

*1: Goto, Tatsumura, & Dixon (2019) に基づきます。

*2: Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.2) の ballistic SB (bSB) に基づきます。

*3: Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.2) の discrete SB (dSB) に基づきます。

3.1.2 sbm_dtype_t

J（スピン間結合係数）行列ビット精度のリストです。

パラメータ

名称	説明
E_SBM_DTYPE_INT1	1bit 符号付き整数（*1）
E_SBM_DTYPE_INT2	2bit 符号付き整数
E_SBM_DTYPE_INT8	8bit 符号付き整数
E_SBM_DTYPE_INT16	16bit 符号付き整数
E_SBM_DTYPE_INT32	32bit 符号付き整数
E_SBM_DTYPE_FP32	単精度（32bit）浮動小数点整数

*1: -1, +1 が設定可能で 0 （結合なし）は設定不可です。全結合型のイジングモデルでのみ使用可能です。

3.1.3 sbm_fpga_id_t

回路インスタンスのリストです。

パラメータ

名称	説明
ISING_B2048FP32_1_0	2048 スピン、SB バージョン B、J（スピン間結合係数）行列ビット精度＝単精度（32bit）浮動小数点整数
ISING_B2048INT1_1_0	2048 スピン、SB バージョン B、J（スピン間結合係数）行列ビット精度＝1bit 符号付き整数

名称	説明
ISING_C2048FP32_1_0	2048 スピン、SB バージョン C、J（スピン間結合係数）行列ビット精度＝単精度（32bit）浮動小数点整数
ISING_C2048INT2_1_0	2048 スピン、SB バージョン C、J（スピン間結合係数）行列ビット精度＝2bit 符号付き整数
ISING_C8192INT2_1_0	8192 スピン、SB バージョン C、J（スピン間結合係数）行列ビット精度＝2bit 符号付き整数

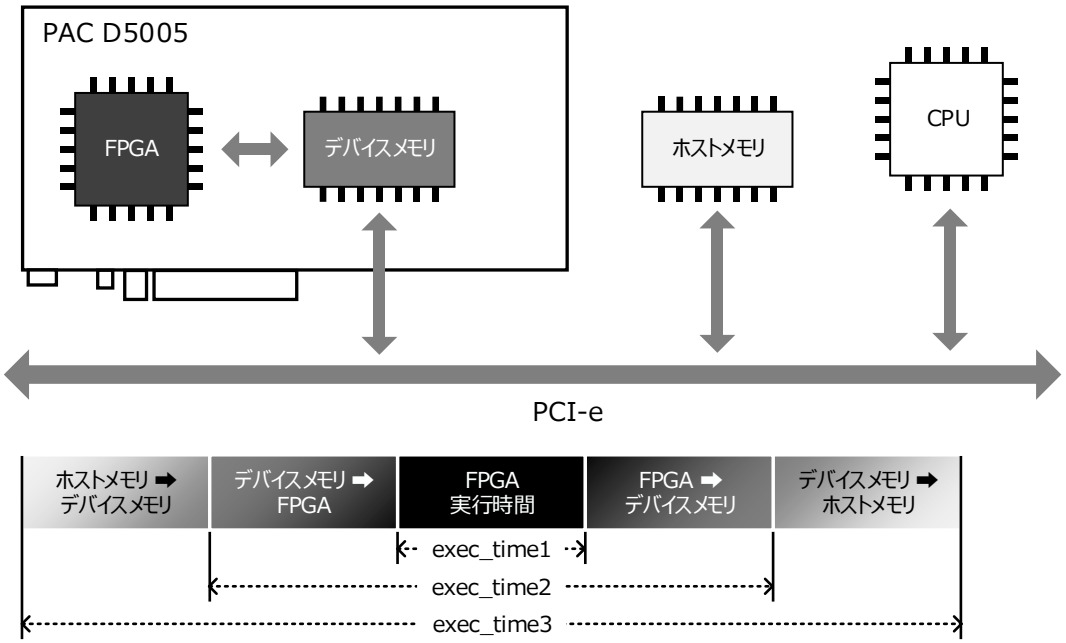
3.1.4 sbm_job_log_t

ジョブハンドル実行ログのリストです。実行時間の内訳は図 3 を参照ください。

パラメータ

名称	型	説明
exec_time1	double	FPGA 実行時間（単位：秒）
exec_time2	double	FPGA 実行時間 ＋ デバイスメモリ転送時間（単位：秒）
exec_time3	double	FPGA 実行時間 ＋ デバイスメモリ転送時間 ＋ ホストメモリ転送時間（単位：秒）

図 3: 実行時間の内訳



3.1.5 H_SBM_FPGA

FPGA ハンドルを定義する型です。

3.1.6 H_SBM_JOB

ジョブハンドルを定義する型です。

3.2 API

3.2.1 sbm_setup_fpga

J 行列精度、最大スピン数、SB アルゴリズムを指定します。条件に合う回路インスタンスをロード・初期化後、呼び出し元に、その回路インスタンスへの FPGA ハンドル（H_SBM_FPGA）を返します。

構文

```
H_SBM_FPGA sbm_setup_fpga(
        sbm_version_t version,
        int max_spin_num,
        sbm_jtype_t jtype
);
```

パラメータ

名称	型	I/O	説明
version	sbm_version_t	I	SB アルゴリズムバージョン
max_spin_num	int	I	最大スピン数
jtype	sbm_jtype_t	I	J 行列ビット精度

戻り値

FPGA ハンドル：正常

NULL：異常

3.2.2 sbm_setup_fpga_id

回路インスタンス型番を指定します。指定の回路インスタンスをロード・初期化後、呼び出し元に、その回路インスタンスへの FPGA ハンドル（H_SBM_FPGA）を返します。

構文

```
H_SBM_FPGA sbm_setup_fpga(
        sbm_fpga_id_t fpga_id
);
```

パラメータ

名称	型	I/O	説明
fpga_id	sbm_fpga_id_t	I	回路インスタンス型番

戻り値

FPGA ハンドル：正常

NULL：異常

3.2.3 sbm_create_job

FPGA ハンドル、スピン数、J 行列、h 行列、制御パラメータを指定します。指定条件に沿ってジョブを生成し、呼び出し元に、そのジョブへのハンドル（H_SBM_JOB）を返します。

構文

```
H_SBM_JOB sbm_create_job(
        H_SBM_FPGA fpga_handle,
        int spin_num,
        float *J,
        float *h,
        int step,
        float c,
        float dt,
        float alph,
        float beta
);
```

パラメータ

名称	型	I/O	説明
fpga_handle	H_SBM_FPGA	I	FPGA ハンドル
spin_num	int	I	スピン数
J	float*	I	J 行列格納先ポインタ。データは Row-major order で配置すること
h	float*	I	h ベクトル格納先ポインタ
step	int	I	SB ステップ数
c	float	I	Goto, Tatsumura, & Dixon (2019, p.6, 式(13)) の ξ_0 に相当

名称	型	I/O	説明
dt	float	I	Goto, Tatsumura, & Dixon (2019, p.6, 式(12),(13)) の Δ_t に相当
alph	float	I	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(7), p.5, 式(12)) の x_i 項に掛かるスケーリング係数（通常は 1.0 を指定）
beta	float	I	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(6), p.5, 式(11)) の y_i 項に掛かるスケーリング係数（通常は 1.0 を指定）

戻り値

ジョブハンドル

イジングモデルの定義

イジングモデルは以下の式で定義されます。パラメータ J、h が式中の J_{ij}、h_i に対応します。

$$C_{Ising} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij}\, s_i s_j + \sum_{i=1}^{N} h_i s_i$$

3.2.4 sbm_set_data

生成済みのジョブに対して、sbm_create_job と同様のパラメータを再設定します。一部パラメータのみの上書きに対応します。

構文

```
void sbm_set_data(
        H_SBM_JOB h_job,
        int spin_num,
        float *J,
        float *h,
        int step,
        float c,
        float dt,
        float alph,
        float beta
);
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル
spin_num	int	I	スピン数
J	float*	I	J 行列格納先ポインタ。データは Row-major order で配置すること。NULL 指定時、無効
h	float*	I	h ベクトル格納先ポインタ。NULL 指定時、無効
step	int	I	SB ステップ数。0 指定時、無効
c	float	I	Goto, Tatsumura, & Dixon (2019, p.6, 式(13)) の ξ_0 に相当
dt	float	I	Goto, Tatsumura, & Dixon (2019, p.6, 式(12),(13)) の Δ_t に相当
alph	float	I	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(7), p.5, 式

名称	型	I/O	説明
			(12)) の x_i 項に掛かるスケーリング係数（通常は 1.0 を指定）
beta	float	I	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(6), p.5, 式(11)) の y_i 項に掛かるスケーリング係数（通常は 1.0 を指定）

戻り値

なし

3.2.5 sbm_set_initial_amplitude_x

生成済みのジョブに対して、位置変数 x（*1）の乱数生成範囲を設定します。位置変数 x は (-1.0, 1.0) の範囲でジョブ実行毎に乱数として生成されます。引数 initial_amplitude によって、この振幅の係数を設定可能です。例えば、initial_amplitude=0.5 としたとき、位置変数 x は (-0.5, 0.5) の範囲で生成されます。2021.3 現在、本 API は回路インスタンス ISING_B2048FP32_1_0, ISING_C2048FP32_1_0 のみで有効です。

*1: Goto, Tatsumura, & Dixon (2019, p.2) の x_i に相当します。

構文

```
void sbm_set_initial_amplitude_x(
        H_SBM_JOB h_job,
        float initial_amplitude
);
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル
initial_amplitude	float	I	乱数生成振幅 [デフォルト=1.0]

戻り値

なし

3.2.6　sbm_set_initial_amplitude_y

生成済みのジョブに対して、運動変数 y（*1）の乱数生成範囲を設定します。運動変数 y は（-1.0, 1.0）の範囲でジョブ実行毎に乱数として生成されます。引数 initial_amplitude によって、この振幅の係数を設定可能です。例えば、initial_amplitude=0.5 としたとき、運動変数 y は（-0.5, 0.5）の範囲で生成されます。

*1: Goto, Tatsumura, & Dixon (2019, p.2) の y_iに相当します。

構文

```
void sbm_set_initial_amplitude_y(
        H_SBM_JOB h_job,
        float initial_amplitude
);
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル
initial_amplitude	float	I	乱数生成振幅 [デフォルト=0.1]

戻り値

なし

3.2.7　sbm_run_sync

指定ジョブを同期型で実行します。

構文

```
sbm_job_log_t sbm_run_sync(
        H_SBM_JOB h_job,
        int *s
);
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル
s	int*	O	スピン行列（ジョブ実行後）格納先ポインタ

戻り値

ジョブハンドル実行ログ

3.2.8 sbm_run_async

指定ジョブを非同期型で実行します。

構文

```
void sbm_run_async(
        H_SBM_JOB h_job
);
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル

戻り値

なし

3.2.9 sbm_get_result

非同期型で実行した指定ジョブの結果を受け取ります。本 API を呼び出した時点で、ジョブが終了していない場合はジョブが終了するまで待機します。

構文

```
sbm_job_log_t sbm_get_result(
        H_SBM_JOB h_job,
        int *s
)
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル
s	int*	O	スピン行列（ジョブ実行後）格納先ポインタ

戻り値

ジョブハンドル実行ログ

3.2.10 sbm_set_random_seed

ランダムパラメータ（位置エネルギー初期値、運動エネルギー初期値）の乱数生成器のシード値を設定します。

構文

```
void sbm_set_random_seed(
        int seed
);
```

パラメータ

名称	型	I/O	説明
seed	int	I	シード値

戻り値

なし

3.2.11 sbm_release_job

指定ジョブのリソースを解放します。

構文

```
void sbm_release_job(
        H_SBM_JOB h_job
);
```

パラメータ

名称	型	I/O	説明
h_job	H_SBM_JOB	I	ジョブハンドル

戻り値

なし

3.2.12 sbm_close

SBMSBM ソフトウェアライブラリで確保した全てのリソースを解放します。

構文

```
void sbm_close();
```

パラメータ

なし

戻り値

なし

3.2.13 sbm_set_message_level

標準出力メッセージのレベルを指定します。

構文

```
void sbm_set_message_level(
        int level
);
```

パラメータ

名称	型	I/O	説明
level	int	I	0：メッセージレベル ERROR 以上を出力 1：メッセージレベル WARN 以上を出力 [デフォルト] 2：メッセージレベル INFO 以上を出力 3：メッセージレベル DEBUG 以上を出力（全てのメッセージを出力）

戻り値

なし

3.2.14 sbm_print_lib_ver

SBM ソフトウェアライブラリのバージョンを出力します。

構文

```
void sbm_print_lib_ver();
```

パラメータ

なし

戻り値

なし

3.2.15 sbm_get_lib_major_ver

SBM ソフトウェアライブラリのメジャーバージョンを受け取ります。

構文

```
int sbm_get_lib_major_ver();
```

パラメータ

なし

戻り値

SBM ソフトウェアライブラリのメジャーバージョン

3.2.16 sbm_get_lib_minor_ver

SBM ソフトウェアライブラリのマイナーバージョンを受け取ります。

構文

```
int sbm_get_lib_minor_ver();
```

パラメータ

なし

戻り値

SBM ソフトウェアライブラリのマイナーバージョン

3.3 エラーメッセージ

SBM がエラーで終了したとき、エラーメッセージが返ります。

表 2: エラーメッセージ一覧

エラーメッセージ	原因・対処方法
Unable to find Intel(R) FPGA OpenCL platform.	OpenCL 環境を正しく参照出来ていません。「2.2.2 ハードウェアのセットアップ」（P.9）を確認してください
Unable to find circuit instance.	該当する回路インスタンスは未サポートです。「第 5 章 回路インスタンス一覧」（P.51）を確認してください
FPGA handler in * is illegal. （*は C API）	FPGA ハンドルの値が不正です。引数を確認してください
JOB handler in * is illegal. （*は C API）	JOB ハンドルの値が不正です。引数を確認してください
Exceed a spin number of circuit instance. arg:spin_num = * / spin number of circuit instance = *. （*はエラー発生時のスピン数）	JOB ハンドルに指定したスピン数が回路インスタンス固有の最大スピン数を超過しています。JOB ハンドルと FPGA ハンドルのスピン数を確認してください
The number of JOB handler reaches an upper limit value 128.	JOB ハンドル数が上限の 128 に達しました。SBM を複数のパラメータで利用したい場合、

	sbm_set_data によるジョブパラメータの再設定、もしくは sbm_relase_job による適宜リリースを推奨します

第4章　Python ライブラリ

Python ライブラリは、C ライブラリと同じ概念で構築されています（図 2）。詳細については、「第 3 章 C ライブラリ」（P.20）を参照ください。

4.1　API

4.1.1　setup_fpga

J 行列精度、最大スピン数、SB アルゴリズムを指定します。条件に合う回路インスタンスをロード・初期化後、呼び出し元に、その回路インスタンスへの FPGA ハンドルを返します。

構文

```
setup_fpga(
        ver,
        max_spin_num,
        jtype
)
```

パラメータ

名称	型	説明
ver	str	SB アルゴリズムバージョン 下記文字列定義を参照ください
max_spin	int	最大スピン数
jtype	str	J 行列ビット精度 下記文字列定義を参照ください

文字列定義

SB アルゴリズムバージョン

文字列	説明
SBM_ALGO_A	SB バージョン A（*1）
SBM_ALGO_B	SB バージョン B（*2）

文字列	説明
SBM_ALGO_C	SB バージョン C（*3）

*1: Goto, Tatsumura, & Dixon (2019) に基づきます。

*2: Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.2) の ballistic SB (bSB) に基づきます。

*3: Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.2) の discrete SB (dSB) に基づきます。

J 行列ビット精度

文字列	説明
SBM_DTYPE_INT1	1bit 符号付き整数（*1）
SBM_DTYPE_INT2	2bit 符号付き整数
SBM_DTYPE_INT8	8bit 符号付き整数
SBM_DTYPE_INT16	16bit 符号付き整数
SBM_DTYPE_INT32	32bit 符号付き整数
SBM_DTYPE_FP32	単精度（32bit）浮動小数点整数

*1: -1, +1 が設定可能で 0 （結合なし）は設定不可です。全結合型のイジングモデルでのみ使用可能です。

戻り値

FPGA ハンドル：正常、 NULL：異常

4.1.2 setup_fpga_id

回路インスタンス型番を指定します。指定の回路インスタンスをロード・初期化後、呼び出し元に、その回路インスタンスへの FPGA ハンドルを返します。

構文

```
setup_fpga(
        fpga_id
)
```

パラメータ

名称	型	説明
fpga_id		回路インスタンス型番

文字列定義

回路インスタンス型番

文字列	説明
ISING_B2048FP32_1_0	2048 スピン、SB バージョン B、J（スピン間結合係数）行列ビット精度＝単精度（32bit）浮動小数点整数
ISING_B2048INT1_1_0	2048 スピン、SB バージョン B、J（スピン間結合係数）行列ビット精度＝1bit 符号付き整数
ISING_C2048FP32_1_0	2048 スピン、SB バージョン C、J（スピン間結合係数）行列ビット精度＝単精度（32bit）浮動小数点整数
ISING_C2048INT2_1_0	2048 スピン、SB バージョン C、J（スピン間結合係数）行列ビット精度＝2bit 符号付き整数
ISING_C8192INT2_1_0	8192 スピン、SB バージョン C、J（スピン間結合係数）行列ビット精度＝2bit 符号付き整数

戻り値

FPGA ハンドル：正常、 NULL：異常

4.1.3 create_job

FPGA ハンドル、スピン数、J 行列、h 行列、制御パラメータを指定します。指定条件に沿ってジョブを生成し、呼び出し元に、そのジョブへのハンドルを返します。

構文

```
create_job(
        fpga_handle,
        spin_num,
        J,
        h,
        step,
        c,
        dt,
        alph,
        beta
)
```

パラメータ

名称	型	説明
fpga_handle		FPGA ハンドル
spin_num	int	スピン数
J	numpy.ndarray.shape(N*N) 又は numpy.ndarray.shape(N,N) dtype=np.float32	J 行列格納先ポインタ。データは Row-major order で配置すること
h	numpy.ndarray.shape(N) dtype=np.float32	h ベクトル格納先ポインタ
step	int	SB ステップ数
c	float	Goto, Tatsumura, & Dixon (2019, p.6, 式(13)) の ξ_0 に相当
dt	float	Goto, Tatsumura, & Dixon (2019, p.6, 式(12),(13)) の Δ_t に相当
alph	float	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(7), p.5, 式

名称	型	説明
		(12)) の x_i 項に掛かるスケーリング係数（通常は 1.0 を指定）
beta	float	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(6), p.5, 式(11)) の y_i 項に掛かるスケーリング係数（通常は 1.0 を指定）

戻り値

ジョブハンドル

イジングモデルの定義

イジングモデルは以下の式で定義されます。パラメータ J、h が式中の J_{ij}、h_i に対応します。

$$C_{Ising} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} J_{ij}\, s_i s_j + \sum_{i=1}^{N} h_i s_i$$

4.1.4 set_data

生成済みのジョブに対して、create_job と同様のパラメータを再設定します。一部パラメータのみの上書きに対応します。

構文

```
set_data(
        h_job,
        spin_num,
        J,
        h,
        step,
        c,
        dt,
        alph,
        beta
)
```

パラメータ

名称	型	説明
h_job		ジョブハンドル
spin_num	int	スピン数
J	numpy.ndarray.shape(N*N) 又は numpy.ndarray.shape(N,N) dtype=np.float32	J 行列格納先ポインタ。データは Row-major order で配置すること
h	numpy.ndarray.shape(N) dtype=np.float32	h ベクトル格納先ポインタ
step	int	SB ステップ数
c	float	Goto, Tatsumura, & Dixon (2019, p.6, 式(13)) の ξ_0 に相当
dt	float	Goto, Tatsumura, & Dixon (2019, p.6, 式(12),(13)) の Δ_t に相当
alph	float	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(7), p.5, 式

名称	型	説明
		(12)) の x_i 項に掛かるスケーリング係数（通常は 1.0 を指定）
beta	float	Goto, Endo, Suzuki, Sakai, Kanao, Hamakawa, Hidaka, Yamasaki, & Tatsumura (2021, p.4, 式(6), p.5, 式(11)) の y_i 項に掛かるスケーリング係数（通常は 1.0 を指定）

戻り値

なし

4.1.5 set_initial_amplitude_x

生成済みのジョブに対して、位置変数 x（*1）の乱数生成範囲を設定します。位置変数 x は (-1.0, 1.0) の範囲でジョブ実行毎に乱数として生成されます。引数 initial_amplitude によって、この振幅の係数を設定可能です。例えば、initial_amplitude=0.5 としたとき、位置変数 x は (-0.5, 0.5) の範囲で生成されます。2021.3 現在、本 API は回路インスタンス ISING_B2048FP32_1_0, ISING_C2048FP32_1_0 のみで有効です。

*1: Goto, Tatsumura, & Dixon (2019, p.2) の x_iに相当します。

構文

```
void set_initial_amplitude_x(
        h_job,
        initial_amplitude
);
```

パラメータ

名称	型	説明
h_job		ジョブハンドル

名称	型	説明
initial_amplitude	float	乱数生成振幅 [デフォルト=1.0]

戻り値

なし

4.1.6 set_initial_amplitude_y

生成済みのジョブに対して、位置変数 y（*1）の乱数生成範囲を設定します。位置変数 y は（-1.0, 1.0）の範囲でジョブ実行毎に乱数として生成されます。引数 initial_amplitude によって、この振幅の係数を設定可能です。例えば、initial_amplitude=0.5 としたとき、位置変数 y は（-0.5, 0.5）の範囲で生成されます。

*1: Goto, Tatsumura, & Dixon (2019, p.2) の y_iに相当します。

構文

```
void set_initial_amplitude_y(
        h_job,
        initial_amplitude
);
```

パラメータ

名称	型	説明
h_job		ジョブハンドル
initial_amplitude	float	乱数生成振幅 [デフォルト=0.1]

戻り値

なし

4.1.7 run_sync

指定ジョブを同期型で実行します。

構文

```
run_sync(
        h_job
)
```

パラメータ

名称	型	説明
h_job		ジョブハンドル

戻り値

スピン行列（ジョブ実行後）格納先ポインタ。型は numpy.ndarray.shape(N), dtype=np.int32 で定義されます。

ジョブハンドル実行ログ。詳細は以下の表と図 3 を参照ください。

ジョブハンドル実行ログ

名称	型	説明
exec_time1	double	FPGA 実行時間（単位：秒）
exec_time2	double	FPGA 実行時間 ＋ デバイスメモリ転送時間（単位：秒）
exec_time3	double	FPGA 実行時間 ＋ デバイスメモリ転送時間 ＋ ホストメモリ転送時間（単位：秒）

4.1.8 run_async

指定ジョブを非同期型で実行します。

構文

```
run_async(
        h_job
)
```

パラメータ

名称	型	説明
h_job		ジョブハンドル

戻り値

なし

4.1.9 get_result

非同期型で実行した指定ジョブの結果を受け取ります。本 API を呼び出した時点で、ジョブが終了していない場合はジョブが終了するまで待機します。

構文

```
get_result(
        h_job
)
```

パラメータ

名称	型	説明
h_job		ジョブハンドル

戻り値

スピン行列（ジョブ実行後）格納先ポインタ。run_sync と同一の戻り値です。

ジョブハンドル実行ログ。run_sync と同一の戻り値です。

4.1.10 set_random_seed

ランダムパラメータ（位置エネルギー初期値、運動エネルギー初期値）の乱数生成器のシード値を設定します。

構文

```
set_random_seed(
        seed
)
```

パラメータ

名称	型	説明
seed	int	シード値

戻り値

なし

4.1.11 release_job

指定ジョブのリソースを解放します。

構文

```
release_job(
        h_job
)
```

パラメータ

名称	型	説明
h_job		ジョブハンドル

戻り値

なし

4.1.12 close

SBM ソフトウェアライブラリで確保した全てのリソースを解放します。

構文

```
void close();
```

パラメータ

なし

戻り値

なし

4.1.13 set_message_level

標準出力メッセージのレベルを指定します。

構文

```
set_message_level(
        level
)
```

パラメータ

名称	型	説明
level	int	0：メッセージレベル ERROR 以上を出力 1：メッセージレベル WARN 以上を出力 [デフォルト] 2：メッセージレベル INFO 以上を出力 3：メッセージレベル DEBUG 以上を出力（全てのメッセージを出力）

戻り値

なし

4.1.14 print_lib_ver

SBM ソフトウェアライブラリのバージョンを出力します。

構文

```
print_lib_ver()
```

パラメータ

なし

戻り値

なし

4.1.15 get_lib_major_ver

SBM ソフトウェアライブラリのメジャーバージョンを受け取ります。

構文

```
get_lib_major_ver()
```

パラメータ

なし

戻り値

SBM ソフトウェアライブラリのメジャーバージョン

4.1.16 get_lib_minor_ver

SBM ソフトウェアライブラリのマイナーバージョンを受け取ります。

構文

```
get_lib_minor_ver()
```

パラメータ

なし

戻り値

SBM ソフトウェアライブラリのマイナーバージョン

4.2 エラーメッセージ

「3.3 エラーメッセージ」（P.35）を参照ください。

第5章　回路インスタンス一覧

2021.3 現在、回路インスタンスのサポート状況は下記の通りです。これらインスタンスのいずれかが対象となるように API をご使用ください。

表 3: 回路インスタンス一覧（プロファイル）

回路インスタンス型番	Algo ver	スピン数	J 行列ビット精度	引数 サポート（*1）			
				h	alph	beta	c
ISING_B2048FP32_1_0	B	2048	FP32	✓	✓	✓	✓
ISING_B2048INT1_1_0	B	2048	INT1				✓
ISING_C2048FP32_1_0	C	2048	FP32	✓	✓	✓	✓
ISING_C2048INT2_1_0	C	2048	INT2				✓
ISING_C8192INT2_1_0	C	8192	INT2				✓

*1: 未サポートの引数を指定した場合、その引数は無視されます。

回路インスタンス毎の FPGA コア演算の所要時間は以下を参照ください。T_{step} は 1SB ステップに要する時間で、T_{step} x SB ステップ数 = FPGA 実行時間（図 3 参照）の関係にあります。

表 4: 回路インスタンス一覧（FPGA コア演算の所要時間）

回路インスタンス型番	T_{step} [usec]
ISING_B2048FP32_1_0	4.301
ISING_B2048INT1_1_0	0.683
ISING_C2048FP32_1_0	4.349
ISING_C2048INT2_1_0	0.359
ISING_C8192INT2_1_0	3.645

第6章　API の使用例

6.1　同期型のサンプルコード［C］

C ライブラリによる同期型のサンプルコードを下記に示します。サンプルコード中で読み込んでいる J 行列ファイルは、 "列座標_行座標_値¥n" のフォーマットを想定しています。なお、簡略化のためエラー処理は省略しています。

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <math.h>
 4 #include "sbm_lib.h"
 5
 6 int main() {
 7         // Problem definition
 8         int spin_num = 2000;
 9
10         // Parameter setting
11         int step = 1000;
12         float c = 0.5/sqrt(spin_num);
13         float dt = 1.25;
14         float alph = 1.0;
15         float beta = 1.0;
16
17         // Prepare J and spin pointer
18         float *J = (float*)malloc(spin_num*spin_num*sizeof(float));
19         int *s = (int*)malloc(spin_num*sizeof(int));
20
21         // Load user data
22         FILE *fp = fopen("../data/K2000_ISING_FORMAT.txt", "r");
23         unsigned int row, col;
24         float value;
25         while (fscanf(fp, "%d %d %f¥n", &row, &col, &value) != EOF){
26                 J[(row-1)*spin_num+col-1] =
27                 J[(col-1)*spin_num+row-1] = value;
28         }
29         fclose(fp);
30
31         // Create FPGA and JOB handle
32         H_SBM_FPGA fh = sbm_setup_fpga(E_SBM_ALGO_C, spin_num,
33                                        E_SBM_DTYPE_INT2);
```

```
34          H_SBM_JOB job = sbm_create_job(fh, spin_num, J, NULL,
35                                         step, c, dt, alph, beta);
36
37          // Run JOB
38          sbm_job_log_t log = sbm_run_sync(job, s);
39          sbm_release_job(job);
40
41          // Dump output
42          printf("FPGA exec time = %.9f / %.9f / %.9f s¥n",
43                  log.exec_time1, log.exec_time2, log.exec_time3);
44          printf("result¥n------¥n");
45          for (unsigned int i=0; i<spin_num; i++){
46                  printf("%d,", s[i]);
47          }
48          printf("¥n");
49
50          free(J);
51          free(s);
52 }
```

サンプルコードの実行結果は以下のようになります。

```
INFO : Time of fpga_setup: 0.093388775 s
FPGA exec time = 0.000330218 / 0.001371643 / 0.001693057 s
result
------
-1,-1,-1,1,-1,1,-1,-1,-1,1,1,-1,1,1,-1,-1,-1,1,1,1,1,-1,-1,-1,1,1,-1,1,-
1,1,1,1,1,1,1,-1,-1,1,-1,-1,1,-1,1,-1,-1,-1,1,1,-1,-1,1,1,
 . . . .
-1,-1,-1,1,-1,1,1,-1,1,1,
```

6.2　非同期型のサンプルコード［C］

C ライブラリによる非同期型のサンプルコードを下記に示します。非同期実行によって、job[1]の FPGA 演算:L55 と job[0]の結果出力:L56 は並列に処理されます。

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 #include <math.h>
 4 #include "sbm_lib.h"
 5
 6 void dump_output(int spin_num, sbm_job_log_t log, int *s){
 7         printf("FPGA exec time = %.9f / %.9f / %.9f s¥n",
 8                 log.exec_time1, log.exec_time2, log.exec_time3);
 9         printf("result¥n------¥n");
10         for (unsigned int i=0; i<spin_num; i++){
11                 printf("%d,", s[i]);
12         }
13         printf("¥n");
14 }
15
16 int main() {
17         // Problem definition
18         int spin_num = 2000;
19
20         // Parameter setting
21         int step[2] = {50, 1000};
22         float c = 0.5/sqrt(spin_num);
23         float dt = 1.25;
24         float alph = 1.0;
25         float beta = 1.0;
26
27         // Prepare J and spin pointer
28         float *J = (float*)malloc(spin_num*spin_num*sizeof(float));
29         int *s = (int*)malloc(spin_num*sizeof(int));
30
31         // Load user data
32         FILE *fp = fopen("../data/K2000_ISING_FORMAT.txt", "r");
33         unsigned int row, col;
34         float value;
35         while (fscanf(fp, "%d %d %f¥n", &row, &col, &value) != EOF){
36                 J[(row-1)*spin_num+col-1] =
37                 J[(col-1)*spin_num+row-1] = value;
```

```
38          }
39          fclose(fp);
40
41          // Prepare JOB
42          H_SBM_JOB job[2];
43
44          // Create FPGA and JOB handle
45          H_SBM_FPGA fh = sbm_setup_fpga(E_SBM_ALGO_C, spin_num,
46                                         E_SBM_DTYPE_INT2);
47          job[0] = sbm_create_job(fh, spin_num, J, NULL,
48                                  step[0], c, dt, alph, beta);
49          job[1] = sbm_create_job(fh, spin_num, J, NULL,
50                                  step[1], c, dt, alph, beta);
51
52          // Run JOB
53          sbm_job_log_t log;
54          log = sbm_run_sync(job[0], s);
55          sbm_run_async(job[1]);
56          dump_output(spin_num, log, s);
57          log = sbm_get_result(job[1], s);
58          dump_output(spin_num, log, s);
59
60          sbm_release_job(job[0]);
61          sbm_release_job(job[1]);
62
63          free(J);
64          free(s);
65 }
```

サンプルコードの実行結果は以下のようになります。

```
INFO : Time of fpga_setup: 0.093725550 s
FPGA exec time = 0.000016511 / 0.001097057 / 0.001455275 s
result
------
-1,1,1,1,-1,1,-1,1,1,-1,-1,-1,-1,-1,-1,1,1,1,1,-1,-1,-1,-1,1,-1,-1,1,-1,-
1,-1,1,-1,-1,-1,1,-1,-1,-1,-1,-1,1,1,-1,1,1,-1,1,-1,-1,-1,
 . . . .
1,1,-1,-1,1,1,1,1,
FPGA exec time = 0.000330218 / 0.001395172 / 0.001714849 s
result
------
1,1,1,1,-1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,1,1,1,1,-1,-1,-1,1,-1,1,-1,-
1,-1,1,-1,-1,1,1,1,-1,1,1,1,1,1,1,1,-1,1,1,1,1,-1,-1,1,1,
 . . . .
1,1,-1,1,-1,1,-1,-1,1,-1,-1,1,-1,1,1,-1,-1,1
```

6.3　同期型のサンプルコード［Python］

Python ライブラリによる同期型のサンプルコードを下記に示します。

```
 1 import pysbm
 2 import pandas as pd
 3 import numpy as np
 4 import math
 5
 6 if __name__ == "__main__":
 7         # Problem definition
 8         spin_num = 2000
 9
10         # Parameter setting
11         step = 1000
12         c = .5/math.sqrt(spin_num)
13         dt = 1.25
14         alph = 1.0
15         beta = 1.0
16
17         # Prepare J pointer
18         J = np.zeros((spin_num, spin_num)).astype(np.float32)
19
20         # Load user data
21         filepath = '../data/K2000_ISING_FORMAT.txt';
22         f = open(filepath)
23         lines = f.readlines()
24         f.close()
25         for line in lines:
26                 ls = line.split()
27                 row, col, val = int(ls[0])-1, int(ls[1])-1, float(ls[2])
28                 J[row, col] = val
29                 J[col, row] = val
30
31         # Create FPGA and JOB handle
32         fh = pysbm.setup_fpga('SBM_ALGO_C', spin_num, 'SBM_DTYPE_INT2')
33         job = pysbm.create_job(fh, spin_num, J, None,
34                                step, c, dt, alph, beta)
35
36         # Run JOB
37         log, s = pysbm.run_sync(job)
38         pysbm.release_job(job)
```

```
39
40          # Dump output
41          print('FPGA exec time = {:.9f} / {:.9f} / {:.9f} s'
42                .format(log.exec_time1, log.exec_time2, log.exec_time3))
43          print('result¥n------')
44          for i in range(0, spin_num):
45                  print('{:d},'.format(s[i]), end="")
46          print('')
```

サンプルコードの実行結果は以下のようになります。

```
INFO : Time of fpga_setup: 0.094472940 s
FPGA exec time = 0.000330218 / 0.001395735 / 0.001717619 s
result
------
-1,-1,-1,1,-1,1,-1,-1,-1,1,1,-1,1,1,-1,-1,-1,1,1,1,1,-1,-1,-1,1,1,-1,1,-
1,1,1,1,1
 ・・・
,1,-1,-1,1,1,1,-1,-1,-1,1,-1,1,1,-1,1,1,
```

6.4　非同期型のサンプルコード［Python］

Python ライブラリによる非同期型のサンプルコードを下記に示します。非同期実行によって、job[1]の FPGA 演算:L51 と job[0]の結果出力:L52 は並列に処理されます。

```
 1 import pysbm
 2 import pandas as pd
 3 import numpy as np
 4 import math
 5
 6 def dump_output(spin_num, log, s):
 7         print('FPGA exec time = {:.9f} / {:.9f} / {:.9f} s'
 8               .format(log.exec_time1, log.exec_time2, log.exec_time3))
 9         print('result¥n------')
10         for i in range(0, spin_num):
11                 print('{:d},'.format(s[i]), end="")
12         print('')
13
14 if __name__ == "__main__":
15         # Problem definition
16         spin_num = 2000
17
18         # Parameter setting
19         step = [50, 1000]
20         c = .5/math.sqrt(spin_num)
21         dt = 1.25
22         alph = 1.0
23         beta = 1.0
24
25         # Prepare J pointer
26         J = np.zeros((spin_num, spin_num)).astype(np.float32)
27
28         # Load user data
29         filepath = '../data/K2000_ISING_FORMAT.txt';
30         f = open(filepath)
31         lines = f.readlines()
32         f.close()
33         for line in lines:
34                 ls = line.split()
35                 row, col, val = int(ls[0])-1, int(ls[1])-1, float(ls[2])
36                 J[row, col] = val
37                 J[col, row] = val
38
39         # Prepare JOB
40         job = [0] * 2;
41
42         # Create FPGA and JOB handle
43         fh = pysbm.setup_fpga('SBM_ALGO_C', spin_num, 'SBM_DTYPE_INT2')
```

```
44          job[0] = pysbm.create_job(fh, spin_num, J, None,
45                                    step[0], c, dt, alph, beta)
46          job[1] = pysbm.create_job(fh, spin_num, J, None,
47                                    step[1], c, dt, alph, beta)
48
49          # Run JOB
50          log, s = pysbm.run_sync(job[0])
51          pysbm.run_async(job[1])
52          dump_output(spin_num, log, s)
53          log, s = pysbm.get_result(job[1])
54          dump_output(spin_num, log, s)
55
56          pysbm.release_job(job[0])
57          pysbm.release_job(job[1])
```

サンプルコードの実行結果は以下のようになります。

```
INFO : Time of fpga_setup: 0.097289678 s
FPGA exec time = 0.000016511 / 0.001059504 / 0.001434496 s
result
------
-1,1,1,1,-1,1,-1,1,1,-1,-1,-1,-1,-1,-1,1,1,1,1,-1,-1,-1,-1,1,-1,-1,1,-1,-
1,-1,1,-1,-1,-1,1,
 . . .
-1,-1,-1,-1,-1,1,-1,-1,1,1,1,
FPGA exec time = 0.000330218 / 0.001399860 / 0.001764015 s
result
------
1,1,1,1,-1,-1,-1,1,-1,1,-1,1,1,-1,1,-1,-1,-1,1,1,1,1,-1,-1,-1,1,-1,1,-1,-
1,-1,1,-1,-1,1,1,1,-1,1,1,1,1,1,1,1,-1,1,1,
 . . .
,1,1,1,-1,-1,1,-1,1,-1,1,1,-1,-1,
```

第7章　リファレンスデザインチュートリアル

7.1　はじめに

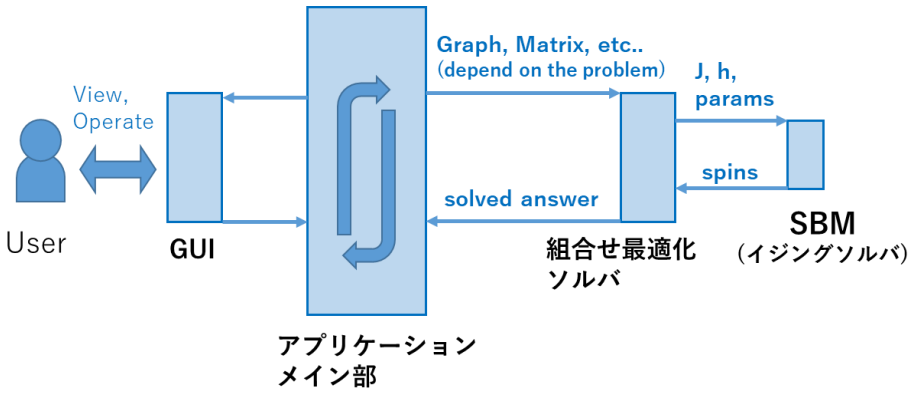
本章では、リファレンスデザインを参照しつつ、実践的なユーザーアプリケーションにおける SBM の組み込み方法を紹介します。東芝が提供するリファレンスデザインは下記のとおりです。

名称	略称	内容
マルチオブジェクトトラッキング	**MOT**	**リアルタイム処理の例。**リアルタイム処理の象徴である動画像処理を対象とし、動画像処理の様子と FPS(Frame Per Sec)を併記しながら FPGA 版 SBM が**リアルタイム処理できること**を示す。動画像処理アプリケーションとしては、MOT (Multi Object Tracking)を引用し、MOT 処理の一部に存在する組合せ最適化問題(最大マッチング問題)に SBM を適用する。
インタラクティブ巡回セールスマン問題ソルバ (**CLI 版**あり)	**TSP**	**インタラクティブアプリの例。**ユーザー操作(周辺状況)に応じ**インタラクティブ**に応答を示すことで、**低レンテンシ性能**を示すアプリケーション。NP-hard の代表的な組合せ最適化問題である TSP (巡回セールスマン問題)を対象とする。
最大独立集合問題ソルバ (**CLI 版**あり)	**MIS**	**高度ストリームデータ処理の例。**高い実用性がありながら NP-hard であり、既存ソルバで大規模問題を効率的に扱えなかった **MIS** (Maximum Independent Set)問題を、高精度・高速に解けることを示す。

以降、本文では上記”名称”の代わりに、”略称”のみを使用する場合があります。

図 4 にリファレンスデザイン(**MOT**、**TSP**、**MIS**)共通のソフトウェア基本構造(Structure)を示します。

図 4: リファレンスデザイン共通のソフトウェア構造



SBM は J 行列と h ベクトルを入力とする「**イジングソルバ**」です。SBM を利用するためには、アプリケーション内で利用する組み合わせ最適化問題をイジング問題に定式化します。また、SBM が算出した解(スピン状態の組み合わせ)が、元の組み合わせ最適化問題の制約条件を満たすか否かを確認する場合もあります。本書では、この定式化処理と制約条件チェックの処理を担当するモジュールを「**組合せ最適化ソルバ**」と呼びます。また、組合せ最適化ソルバ以外のアプリケーション全体処理を担うモジュールを総括して「**アプリケーションメイン部**」と呼ぶことにします。**MOT**、**TSP**、**MIS** はいずれもグラフィカルユーザーインタフェース「**GUI**」を備えていますが、**TSP** と **MIS** については GUI なしで動作する **CLI**(Command Line Interface)版も用意しています。

各デザイン、各モジュールをそれぞれ異なるプログラミング言語で設計しています。モジュールと設計言語の対応関係については下表を参照してください。SBM については、C/C++版 API と Python 版 API を用意していますが、今回提供するリファレンスデザインはいずれも C/C++版 API を使用しています。

		GUI	アプリケーションメイン部	組合せ最適化ソルバ	SBM
MOT		Python (*2)	Python	C/C++ (*1)	C/C++ SBM API
TSP	CLI	N/A	C/C++	C/C++	
		JavaScript Python (*3)	Python	C/C++ (*1)	
MIS	CLI	N/A	Python	C/C++ (*1)	
		JavaScript Python (*3)			

(*1) C/C++で記述した組合せ最適化ソルバをダイナミックリンクライブラリ(*.so)としてビルド。C/C++ライブラリをロード&wrap し、Python コードから利用できる、組合せ最適化ソルバ固有の Python 向け function を用意

(*2) GUI フレームワークとしては、PySimpleGUI、OpenCV を使用

(*3) GUI は、Web アプリとして実装。Web アプリフレームワークとして Responder を使用

それぞれのリファレンスデザインは、下記の場所に存在します。

MOT: $examples/mot
TSP: $examples/tsp (CLI および GUI 向けモジュール)
MIS: $examples/mis (CLI および GUI 向けモジュール)
TSP と MIS の共通 GUI 環境: $examples/gui_server

本章における表記**$examples** はリファレンスデザインのトップディレクトリを指し、2.3.1 で設定した**$SBM_HOME** を用いると**$SBM_HOME/examples** に該当します。必要に応じ環境変数として設定ください。以下は Bash シェル利用時の環境変数設定例です。

```
$ export examples=${SBM_HOME}/examples
```

Python 環境について

本チュートリアルは、**Python3.6** 環境を対象としています。以降の説明においては、Python はシステムにインストールされた Python(**python3**)を使用し、pip(**pip3**)による外部パッケージのインストール先はユーザーディレクト内である(**--user** オプション付き)ことを前提(※)に実行例を示します。また、次節以降の手順を進める前に、以下のように pip のアップグレードを行うことを推奨します。

```
$ pip3 install -U pip --user
```

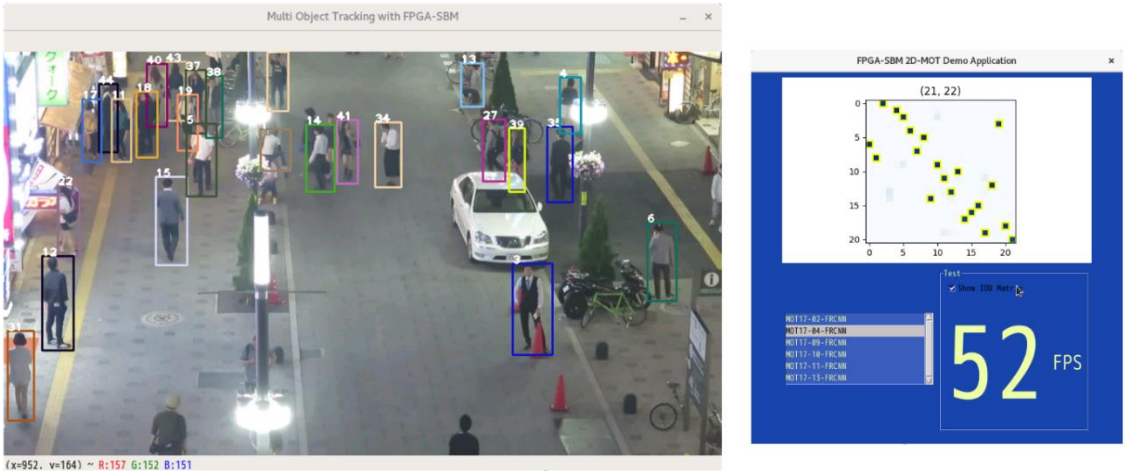
(※) ユーザーの Pyhon 環境が本前提と異なる場合は、以降の説明文はユーザー環境に合わせ適宜読み替えてください。

7.2　マルチオブジェクトトラッキング

7.2.1　概要

MOT（Multi Object Tracking）は、動画内の複数の物体をそれぞれ区別しながら（特定して）追跡する技術です。追跡対象が「人物」である場合は、動画フレームの中から「人物」の特定をし、ID の割り当てを行います。ここでは、MOT アルゴリズムとして既存手法である **SORT**[2]を利用し、その組合せ最適化問題部に SBM を適用します。本アプリケーションでは、GitHub で公開されている Python スクリプト[3]を SORT モジュールとして使用します。

図 5: MOT 動作イメージ



7.2.2　二部グラフ最大マッチング問題

MOT（SORT）における 1 フレームあたりの実施処理は大きく以下の 2 者に分類できます。

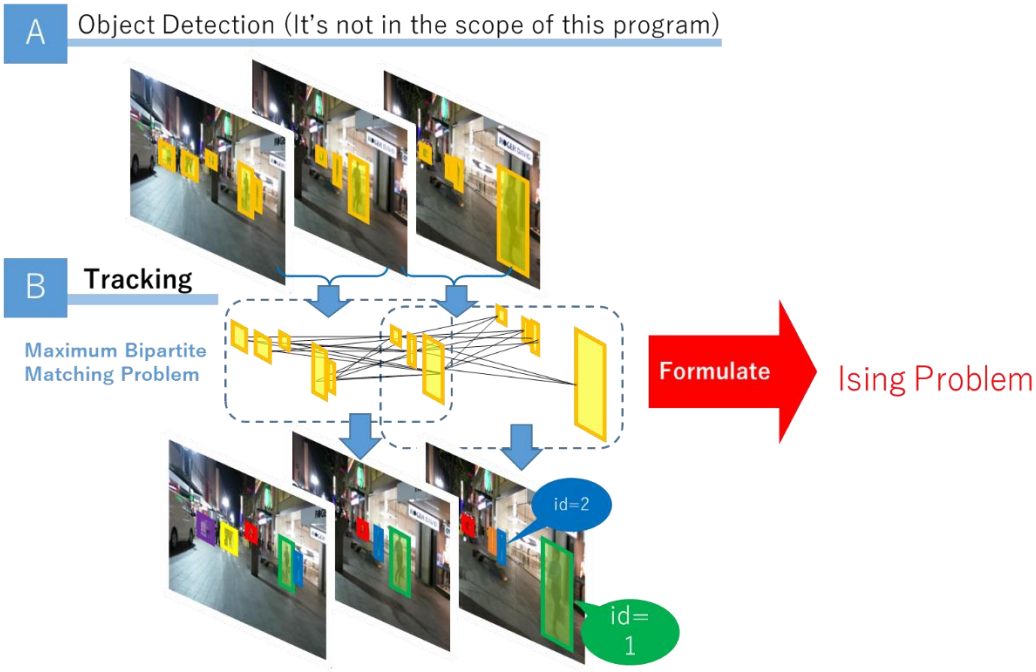
A. オブジェクト検出(+認識、位置推測)
B. トラッキング(検出済みオブジェクトへの ID 割り当て)

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, Ben Upcroft, Simple Online and Realtime Tracking, https://arxiv.org/abs/1602.00763

[3] https://github.com/abewley/sort

ここで、B の「トラッキング」の一部処理は、二部グラフ最大マッチング問題と呼ばれる組み合わせ最適化問題に帰着されます。本デザインでは、この組み合わせ最適化問題を、イジング問題に定式化して SBM で解きます。

図 6: MOT 構成要素と組合せ最適化問題



7.2.3 ファイル構成

関連ファイル($examples 基点の相対パス表記)を列挙します。

ディレクトリ	ファイル名	内容
mot	mot.py	アプリケーションメイン Python スクリプト
mot	mbmsbm.py	最大マッチング問題ソルバの Python wrapper
mot	mkvideo.sh	表示用動画ファイルの生成スクリプト
mot/mbmsbmlib	mbmsbm.cpp	最大マッチング問題ソルバの C++実装
mot/mbmsbmlib	Makefile	mbmsbm.cpp から 共有ライブラリ mbmsbmlib.so を 生成するための Makefile
mot	requirements.txt	Python モジュールリスト

東芝から提供する上記ファイル以外に、データセットの準備が必要になります(次節で説明)。

7.2.4　実行環境の準備

Python パッケージのインストール

本デザイン実行に必要な Python パッケージをインストールします。

```
$ pip3 install -r requirements.txt --user
```

SORT スクリプトの準備

本デザインで使用する SORT スクリプトをダウンロード(git clone)し、本デザインで動作確認ができているコミットをチェックアウトします。下記実行には git が必要です(事前に git をシステムにインストールしてください)。

```
$ cd $examples/mot
$ git clone https://github.com/abewley/sort.git sort
$ cd sort
$ git checkout bce9f0d1fc8fb5f45bf7084130248561a3d42f31
```

データセットの準備

本デザインは、公開データセット"MOT Challenge" (https://motchallenge.net/)のうち、MOT17の一部を使用します。

1.ダウンロード

https://motchallenge.net/data/MOT17/

→ Download, Get all data (5.5GB) : https://motchallenge.net/data/MOT17.zip

から MOT17.zip をダウンロードし、zip 解凍したのち適当なディレクトリに展開します。次項説明のため、この展開先ディレクトリのパスを<MOT17 directory>と表記します。

2.シンボリックリンクの生成

データセットに対するシンボリックリンクを生成します。

```
$ cd $examples/mot
$ ln -s <MOT17 directory> ./
```

3.動画ファイル生成

MOT 実行ディレクトリ下で mkvideo.sh を実行し、jpg ファイルから動画ファイル(*.mp4)を生成します。本スクリプト実行には ffmpeg が必要です(事前に ffmpeg をシステムにインストールしてください)。

```
$ cd $examples/mot
$ ./mkvideo.sh
```

プログラムのビルド

本デザインでは、組み合わせ最適化ソルバを C/C++で実装しています。組み合わせ最適化ソルバは、Python スクリプト実行時にダイナミックリンクライブラリとしてロードされますが、事前にビルドが必要です。

```
$ cd $examples/mot/mbmsbmlib
$ make
```

本 operation により、$examples/mot/mbmsbmlib/以下に **libmbmsbm.so** が生成されます

7.2.5　実行方法

実行手順を示します。

```
$ cd $examples/mot
$ python3 mot.py
```

図 5 のように、2 つのウィンドウが起動します。左側のウィンドウでは、トラッキング結果を重畳した動画を示します。右側のウィンドウ(“制御ウィンドウ”と呼びます)は、動画シーケンスの切り替え機能や、FPS (Frame Per Second)表示機能を提供[4]します。

[4] 図 5 では、前後フレーム IOU 間の最大マッチング結果も描画されていますが、現在のバージョンでは本機能は動作しません(“Show IOU Matrix”チェックボックス機能が無効)。本機能を有効にするには、sort/sort.py と mot.py の一部に編集が必要です(本書では言及しません)。

終了する際は、mot.py を実行したターミナルで“Ctrl + C”を押下するか、制御ウィンドウ右上部の”x”を押下してください。

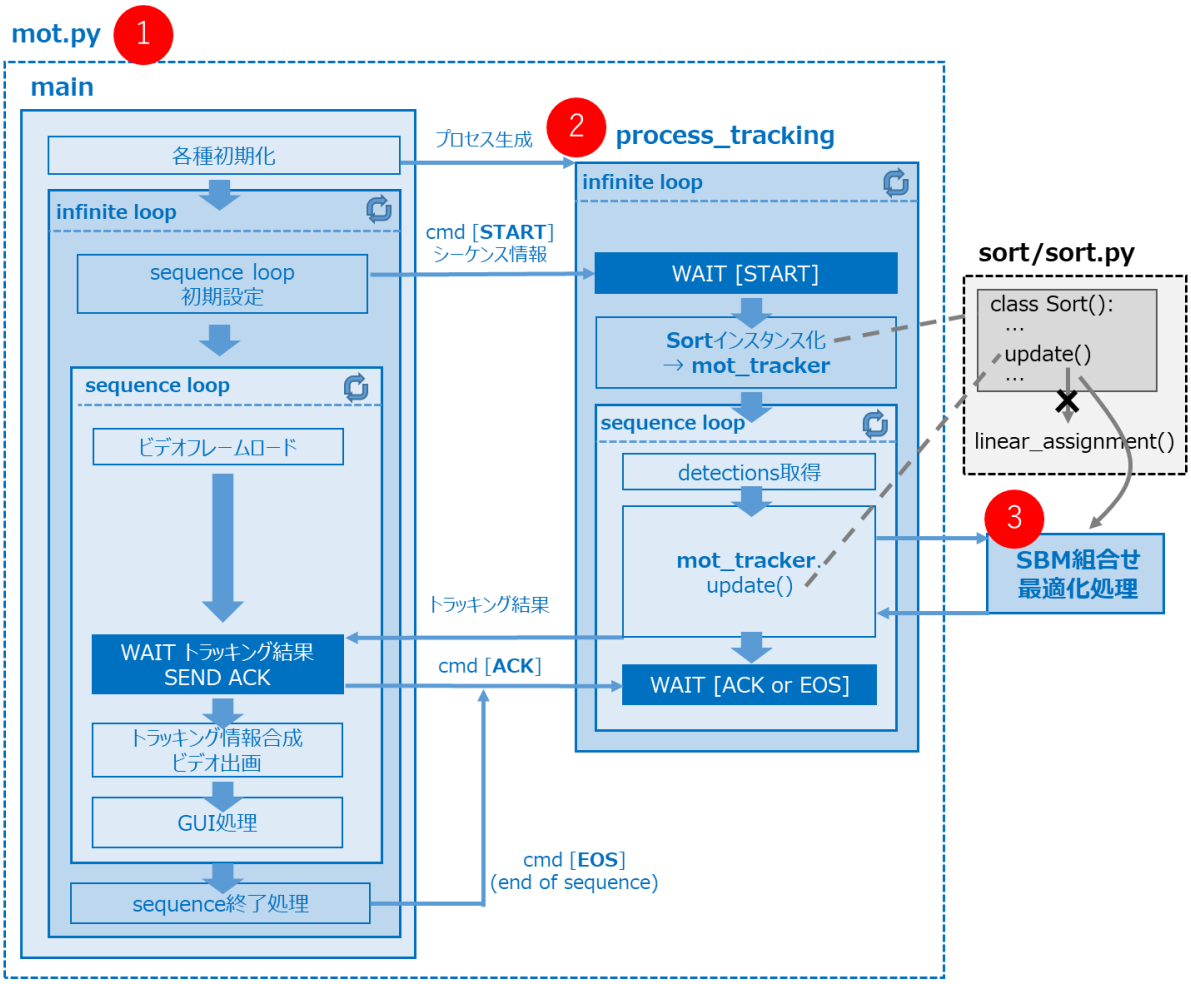
7.2.6 解説

MOT アプリケーションの処理フロー概念図を図 7 に示します。ここでは、

- mot.py 全体概要
- process_tracking 処理
- SBM 組合せ最適化処理

の 3 項目で、その実装内容を説明します。

図 7：MOT アプリケーション処理フロー概念



1. mot.py 全体概要

MOT アプリケーションは大きく 2 つのプロセスで構成されます。1 つ目が **main** プロセスで、2 つ目が **process_tracking** プロセスです。**main** プロセスは、主にビデオ処理(フレームのロード、トラッキング結果をフレームに重畳、ビデオ出力処理)と GUI 処理、およびプログラム全体の状態管理を担当します。一方、**process_tracking** プロセスは MOT の主機能であるトラッキング処理を担当します。

main プロセスは、その初期段階で下記のように **process_tracking** プロセスを生成します。

mot.py

```
  9 import multiprocessing as mp
-------------- 中略 --------------
127     # Queues for communication between main and tracking process
128     q_cmd   = mp.Queue()
129     q_track = mp.Queue()
130
131     # Launch sub process for tracking
132     p = mp.Process(target=process_tracking, args=(q_cmd, q_track))
133     p.start()
```

main プロセスと **process_tracking** プロセスは、上記に示す 2 つの Queue を介して相互通信します。ここで、q_cmd は **main** プロセスから **process_tracking** プロセス方向、q_track は逆方向の通信に使用します(**process_tracking** 内ではそれぞれ q_in、q_out と表現されます)。

main プロセスは、**process_tracking** プロセス生成後、GUI で指定された sequence(一つの動画を構成するフレーム画像の集合)を選び、sequence 固有の初期設定を行います。その後、選択された sequence の全てのフレームに対して、ビデオフレームロードからビデオ出画までの一連のビデオ処理を実施します(sequence loop)。その過程で、**process_trakcking** で計算されたトラッキング結果を q_track 経由で受信し、出画フレームへの重畳を行います。また、1 フレーム単位で GUI やシステム状態を確認し、必要に応じ sequence 切り替えやプログラム終了の判定をします。

2. process_tracking 処理

以下に、**process_tracking** プロセスの Python 実装を示します。

mot.py

```
  6 import sort.sort as sort
-------------- 中略 --------------
```

```
45 def process_tracking(q_in, q_out):
46     """
47     Main tracking procedure.
48     This function should be run on another process from main.
49     q_in: Queue for commands to this process
50     q_out: Queue for output data of tracking result
51     """
52 
53     while True:
54 
55         # Wait for 'start' command, and get 'seq_dets'
56         while True:
57             idata =  q_in.get()
58             if idata[0] == 'start':
59                 seq_dets = idata[1]
60                 break
61 
62         # Instantiate Sort object as mot_tracker
63         mot_tracker = sort.Sort()
64         sort.KalmanBoxTracker.count = 0
65 
66         for frame in range(int(seq_dets[:,0].max())):
67             frame += 1
68 
69             dets = seq_dets[seq_dets[:,0]==frame,2:7]
70             dets[:,2:4] += dets[:,0:2]
71 
72             trackers = mot_tracker.update(dets)
73 
74             q_out.put([trackers, None])
75 
76             idata =  q_in.get() # wait ack
77             if idata[0] == 'end':
78                 break
```

本アプリケーションでは、7.2.4 でユーザーがダウンロードする sort/sort.py を import(L6)し、外部モジュールとして利用します。

main プロセスで新たな sequence が選択される度に、'start'コマンドとともに sequence 内の全 bounding box 情報(seq_dets)が q_in に格納されます。これらの情報を得た後、sort.py 内で定義される Sort クラスのオブジェクトをインスタンス化(mot_tracker)し、sequence loop に入ります。

sequence loop 内では、当該フレーム番号における検出情報(bounding boxes)を取得し(dets)、Sort オブジェクト(mot_tracker)の update()を呼び出すことで、トラッキング処理を実行します。トラッキング実行結果(trackers)は q_out を介して **main** プロセスに渡します。

3. SBM 組合せ最適化処理

Sort クラスの update()内の処理の一部として、組合せ最適化問題である二部グラフ最大マッチング問題を解くプロセスがあり、オリジナルの sort.py では、linear_assignment()関数がこれに相当します。本 MOT 例では、linear_assignment()の代わりに SBM で実装した二部グラフ最大マッチング問題ソルバを適用します。ここでは、

A.SBM による二部グラフ最大マッチング問題ソルバ(C++)を実装する手順

B. A のソルバを Python から呼び出す手順

C.linear_assignment()の代わりに B のソルバを呼び出す手順

の順で解説します。

A.SBM による二部グラフ最大マッチング問題ソルバ(C++)を実装する手順

mbmsbmlib/mbmsbm.cpp から一部抜粋して説明します。

最初に、二部グラフ最大マッチング問題とイジング形式間の関係を取り持つ機能群を、一つのクラス(MaximuBipartileMatching)として定義します(下記参照)。

mbmsbmlib/mbmsbm.cpp

```
 23 class MaximumBipartiteMatching{
 24     private:
 25     int n1, n2, n_min;
 26     float *wm, *Q;
 27     float *wm_norm = NULL;
 28
 29     public:
 30     int N;
 31     float *J, *h;
 32
-------------- 中略 --------------
 87     /**
 88      * Formulate MBM(weight:n1*n2) to Q(N*N), w/ constraint coeff A
 89      */
 90     void formulate_mbm2qubo(float *weight, float A, float B){
-------------- 中略 --------------
160     }
161
162     /**
163      * Convert format from QUBO(Q) to Ising(J,h)
164      */
165     void qubo2ising(){
```

```
166         for(int i=0;i<N;i++){
167             J[N*i+i] = 0;
168             h[i] = 0.5*Q[N*i+i];
169             for(int j=0;j<N;j++){
170                 if(j!=i){
171                     J[N*i+j] = -0.5*Q[N*i+j];
172                     h[i] += 0.5*Q[N*i+j];
173                 }
174             }
175         }
176     }
177 
178     /**
179      * Generate Ising input data
180      */
181     void generate_ising(float A, float B, bool norm=false){
182         if(norm){
183             normalize_wm();
184             formulate_mbm2qubo(wm_norm, A, B); // wm_norm -> Q
185         }else{
186             formulate_mbm2qubo(wm, A, B); // wm -> Q
187         }
188         qubo2ising(); // Q -> J,h
189     }
190 
191     /**
192      * Convert spin data to MBM answer
193      */
194     int conv_spin2answer(int *s, int *ans){
195 
196         int cnt = 0;
197         for(int i=0; i<n1; i++){
198             for(int j=0; j<n2; j++){
199                 if(s[i*n2+j] > 0){
200                     if(cnt >= n_min){
201                         return 1;
202                     }
203                     ans[cnt*2]   = i;
204                     ans[cnt*2+1] = j;
205                     cnt++;
206                     //printf("(%d %d)¥n", i, j);
207                 }
208             }
209         }
210         return 0;
211     }
212 };
```

本実装例では、元の組合せ最適化問題(original)からイジング形式(ising)への定式化を、QUBO 形式を中間状態として original→QUBO、QUBO→ising のように 2 段階で行います。

formulate_mbm2qubo()[L90]が、元の組合せ最適化問題から QUBO 形式への定式化を実装する関数です。ここで二部グラフの重み行列(wm[L26] or wm_norm[L27])から QUBO 行列(Q[L26])への変換を行います。

qubo2ising()[L165]は、QUBO 形式から SBM が扱えるイジング形式への変換を実装する関数です。ここで QUBO 行列(Q[L26])から J 行列(J[L31])、h ベクトル(h[L31])への変換を行います。

generate_ising()[L181]は、便宜上用意したラッパ関数で、内部で formulate_mbm2qubo()と qubo2ising()を呼び出します。

conv_spin2answer()[L194]は、イジングソルバの解であるスピンベクトル形式から二部グラフ最大マッチング問題が要求する解形式への変換を行う関数です。ここでは、±1 の 2 値を要素とするスピンベクトル(s[L194])から、マッチングにより 1 対 1 で対応するノードの組(A_n, B_n)をメモリ上で連続配置する形式で変換します。

次に、一連の SBM API による SBM イジングソルバへのアクセス機能をまとめたユーティリティ関数 **sbm_ising_solver()**(※)を実装します。

mbmsbmlib/mbmsbm.cpp

```
  8 #include "sbm_lib.h"
-------------- 中略 --------------
 13 #define NSPIN 2048
 14
 15 // Global variables
 16 static H_SBM_FPGA fh = NULL;
 17 static H_SBM_JOB fjob = NULL;
 18 static int s[NSPIN];
-------------- 中略 --------------
214 /**
215  * SBM Ising solver
216  */
217 extern "C" void sbm_ising_solver(int N, float *J, float *h, int Ncycle, float
c,
218     float dt, int *s){
219
220     sbm_set_random_seed(1);
221
222     if(fjob != NULL){
223         sbm_set_data(fjob, N, J, h, Ncycle, c, dt, 1.0, 1.0);
224     }else{
```

```
225         fh = sbm_setup_fpga(E_SBM_ALGO_B, NSPIN, E_SBM_DTYPE_FP32);
226         fjob = sbm_create_job(fh, N, J, h, Ncycle, c, dt, 1.0, 1.0);
227     }
228     sbm_run_sync(fjob, s);
229 }
```

本関数は、初回呼び出し時に **sbm_setup_fpga()**により FPGA ハンドル(fh[L16])、**sbm_create_job()**により 1 つのジョブ(fjob[L17])を作成し、2 回目以降の呼び出しに対しては作成済みのジョブ(fjob)に対し **sbm_set_data()**でパラメータ設定を行います。

ジョブ設定後、**sbm_run_sync()**により同期実行が行われ、変数 s に解が格納されます。

(※)本例では、sbm_ising_solver()のようなユーティリティ関数を用意しましたが、全てのアプリケーションで必要になるわけではありません。

最後に、Python スクリプト側とのインタフェースとなる関数 **mbmsbm()** を実装します。

mbmsbmlib/mbmsbm.cpp

```
231 /**
232  * SBM MBM solver
233  */
234 extern "C" void mbmsbm(const int n1, const int n2, float *wm, int Ncycle,
235     float c, float dt, int *ans){
236
237     MaximumBipartiteMatching mbm = MaximumBipartiteMatching(n1, n2, wm);
238     mbm.generate_ising(0.5, 0.5, false);
239
240     sbm_ising_solver(mbm.N, mbm.J, mbm.h, Ncycle, c, dt, s);
241     mbm.conv_spin2answer(s, ans);
242 }
```

本関数では、 二部グラフ重み行列の入力・イジング形式への定式化・SBM イジングマシンの実行・解の返答、までの一連のフローを、これまで説明してきた関数を使って実装しています。

Python 側ではこの **mbmsbm()** のみを使用します。

mbmsbmlib/Makefile を使って mbmsbmlib/mbmsbm.cpp をビルドすると、ダイナミックリンクライブラリ mbmsbmlib/libmbmsbm.so が生成されます。

B. A のソルバを Python から呼び出す手順

ここでは、mbmsbm.py(mbmsbm モジュール)内で、Python 版 mbmsbm 関数を定義します。

mbmsbm.py

```
 1 import numpy as np
 2 import numpy.ctypeslib as npct
 3 from ctypes import *
 4
 5 lib = npct.load_library('libmbmsbm.so', './mbmsbmlib')
 6 c_af1 = npct.ndpointer(dtype=np.float32, ndim=1, flags='C_CONTIGUOUS')
 7 c_ai1 = npct.ndpointer(dtype=np.int32, ndim=1, flags='C_CONTIGUOUS')
 8
 9 lib.mbmsbm.argtypes = [c_int, c_int, c_af1, c_int, c_float, c_float, c_ai1]
10 lib.mbmsbm.restype = None
11
12 def mbmsbm(wm, Ncycle, c, dt):
13   n1,n2 = wm.shape[0],wm.shape[1]
14   nmin = min(n1,n2)
15   ans = np.zeros(nmin*2, dtype=np.int32)
16   lib.mbmsbm(n1, n2, wm.flatten().astype(np.float32), Ncycle, c, dt, ans)
17   return ans.reshape([nmin, 2])
```

最初に、lib オブジェクトに A で作成したダイナミックリンクライブラリ(libmbmsbm.so)をロードし、lib.mbmsbm の引数と返り値の型を指定します[L5-L10]。

次に、lib.mbmsbm をラップする python 関数 **mbmsbm()**[L12-L17]を定義します。ここでは、python 側と lib.mbmsbm の間でデータ型の整合性を合わせながら lib.mbmsbm を呼び出すよう実装します。

C. linear_assignment()の代わりに B のソルバを呼び出す手順

ここでは、オリジナルの sort.py 中に実装され、Sort.update()中で呼ばれる linear_assignment()関数の代わりに SBM で実装したソルバを呼び出す方法を示します。

mot.py

```
 6 import sort.sort as sort
 7 import mbmsbm
-------------- 中略 --------------
17 def sbm_mbm_solver(iou_matrix):
18     """
19     Alternative function to sort.linear_assignment
20     (Wrapper of mbmsbm.mbmsbm)
21     """
22     return mbmsbm.mbmsbm(-iou_matrix, 1000, 0.4, 0.4)
23
24 sort.linear_assignment = sbm_mbm_solver;
```

最初に sort モジュールと mbmsbm モジュールを import します(L6-L7)。

オリジナルの **linear_assignment()**関数と同等の引数/返り値を持つような **mbmsbm()**関数のラッパ関数 **sbm_mbm_solver()**を定義します(L17-L22)。

sort.linear_assignment の参照先を sbm_mbm_solver で上書きします(L24)。

これにより、linear_assignment()関数が呼ばれるタイミングで、linear_assignment()の代わりに sbm_mbm_solver()関数が呼ばれます。

7.3　インタラクティブ巡回セールスマン問題ソルバ

7.3.1　概要

TSP（Traveling Salesman Problem）は、都市の集合と各 2 都市間の距離が与えられたとき、全ての都市を 1 度ずつ訪れたうえで出発地に戻る巡回路のうち総距離が最小である経路を求める、という代表的な組み合わせ最適化問題です。本デザインは、Dantzig42 という 42 都市を対象にした一般的なベンチマーク問題をベースに、GUI からのユーザー操作による各都市位置の再配置に対しインタラクティブに経路探索を実行することのできるアプリケーションプログラムの実装例です。GUI 全体イメージと、ユーザー操作(マウスによる都市位置のドラッグ&ドロップ)イメージをそれぞれ図 8、図 9 に示します。

図 8：TSP GUI 全体イメージ

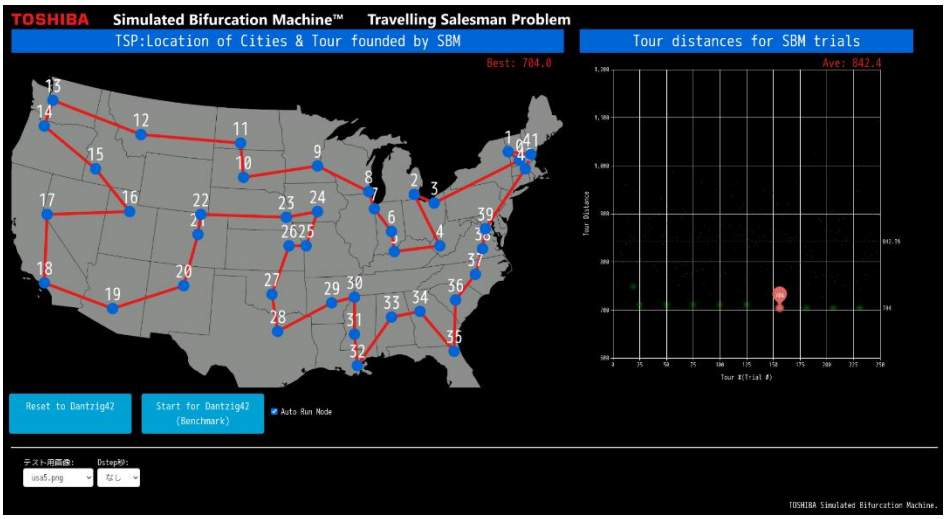
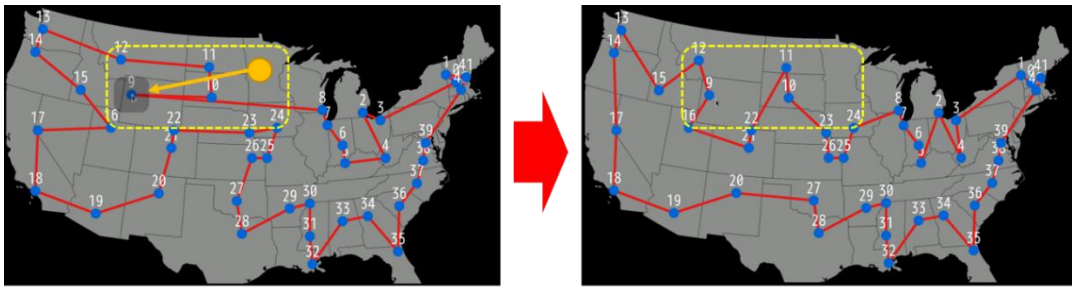


図 9：TSP ユーザー操作イメージ



(a) 9番をマウスでドラッグ中　　(b) ドロップでリルート

7.3.2　ファイル構成

SBM TSP ソルバおよび CLI 版に関連するファイル($examples 基点の相対パス表記)を列挙します。

ディレクトリ	ファイル名	内容
tsp	main.cpp	TSP ソルバの C++実装
tsp	Makefile	main.cpp からスタンドアローン型実行ファイル tsp42.elf とダイナミックリンクライブラリ libsbmtsp42.so を生成するための Makefile
tsp	format_dantzig42.py	dantzig42.tsp から dantzig42.txt (tsp42.elf で読み込み可能な形式)を生成する Python スクリプト

GUI 版の関連ファイルは多数であるため、主要なものの概要のみを下記に示します。必要に応じ gui_server 以下のファイルを参照ください。

ディレクトリ	ファイル名	内容
gui_server	run.py	GUI サーバー実行スクリプト
gui_server	requirements.txt	Python モジュールリスト
gui_server/py	*.py	GUI 版 TSP・MIS 関連 Python ファイル
gui_server/static	*.js	GUI 版 TSP・MIS 関連 JavaScript ファイル

7.3.3　実行環境の準備（CLI 版・GUI 版共通）

データセットの準備

本デザインは、公開データセット"Dantzig42"を使用します。既に「2.3.2 診断プログラム」（P.18）を実行済みの場合、必要なデータは準備できています。下記のデータが存在することを確認してください。もし存在しなければ「2.3.2 診断プログラム」（P.18）の手順を実施ください。

ディレクトリ	ファイル名
$examples/tsp	dantzig42.txt、dantzig42.tsp
$examples/gui_server/data	dantzig42.tsp

プログラムのビルド

C/C++で実装された CLI 版および GUI 版向けの SBM-TSP ソルバをビルドします。

```
$ cd $examples/tsp
$ make
```

これにより、$examples/tsp 以下に **tsp42.elf** (CLI 版実行ファイル)、**libsbmtsp42.so** (GUI 版向けダイナミックリンクライブラリ)が生成されます。

7.3.4 実行環境の準備 (GUI 版向け)

Python パッケージのインストール

本デザイン実行に必要な Python パッケージをインストールします。

```
$ cd $examples/gui_server
$ pip3 install -r requirements.txt --user
```

Node.js と Yarn のインストール

本デザインの実行・再ビルドには **Node.js**(確認済バージョン:12.18.4)と **Yarn**(確認済バージョン:1.22.5)が必要です。以下にそれらのインストール例を示します。

```
# スーパーユーザでの実行を推奨します
$ su
$ curl --silent --location https://dl.yarnpkg.com/rpm/yarn.repo | tee /etc/yum.repos.d/yarn.repo
# Node.js 12.x 系のインストール
$ curl --silent --location https://rpm.nodesource.com/setup_12.x | bash -
$ yum clean all
$ yum install nodejs
# Yarn のインストール
$ yum install yarn
```

javascript パッケージのインストール

前節の **Node.js**、**Yarn** のインストールを済ませた上で下記を実行してください。

```
$ cd $examples/gui_server
$ yarn install
$ yarn build
```

MIS ソルバライブラリのビルド

現在の GUI 実行プログラムは、動作対象が TSP・MIS に依らず MIS ソルバのライブラリを使用します。あらかじめ 7.4.4 の「プログラムのビルド」を実施してください。

7.3.5　実行方法

CLI 版

TSP42 都市問題 dantzig42 を、SBM で解きます。SBM 動作条件は下記のとおりです。

- SBM アルゴリズム: E_SBM_ALGO_B
- J 行列精度: E_SBM_DTYPE_FP32
- [step, c, dt, alph, beta]: 1000, 0.5, 0.1, 0.5, 2.0

上記設定で 200 回実行し、制約条件を満たす解のうちその最小距離と求まったルートを表示します。

```
$ cd $examples/tsp
$ ./tsp42.elf
```

実行結果は下記のようになります。

```
INFO : Time of fpga_setup: 0.150052986 s
minimum distance: 704.00
The route is as follows
[ 16 15 14 13 12 11 10 9 8 7 6 5 4 2 3 0 1 41 40 39 38 37 36 35 34 33 32
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 ]
```

GUI 版

GUI 版は、GUI としてブラウザを使用します。まず、下記のように run.py を実行し、ローカルサーバを起動します。

```
$ cd $examples/gui_server
$ python3 run.py
```

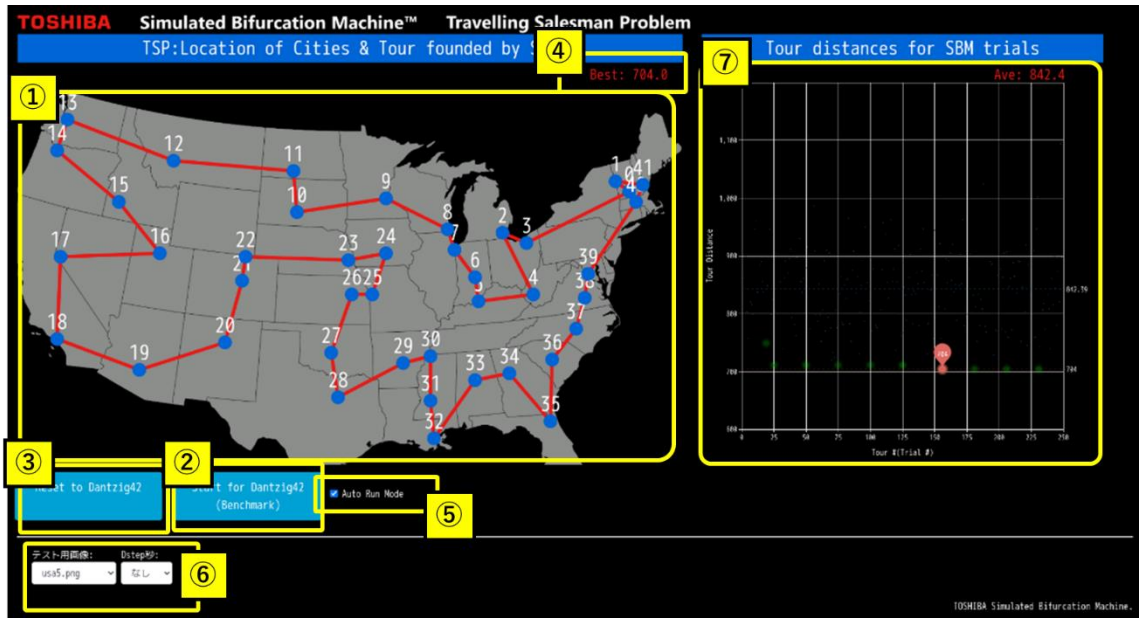
正常に起動されると、下記のようにコンソールに URL が表示されます。

```
INFO:     Started server process [6933]
INFO:     Uvicorn running on http://127.0.0.1:5042 (Press CTRL+C to quit)
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

上記の場合、ブラウザで「**http://127.0.0.1:5042/tsp**」にアクセスしてください。正常動作している場合は、図 8 のような画面が表示されます。

ここでは、TSP GUI 画面について図 10 を参照しながら説明します。

図 10：TSP GUI 画面項目説明



① **[メイン画面]** 都市の配置と経路を図示します。マウスのドラッグ&ドロップで各都市の位置を変更することができます。都市位置を手動で変更した際の挙動は⑤の”Auto Run Mode”がチェック状態により異なります。

② **[Start ボタン]** 本ボタン色が反転されていない場合(Gray ではなく Blue 色状態)、押下することで経路探索を開始します。(プログラムが初期状態[Dantzig42 配置]かそうでないかでボタンのラベルが異なります)

③ **[Reset to Dantzig42 ボタン]** 手動で都市配置を変更した後、本ボタンを押下することで初期状態[Dantzig42 配置]に戻すことができます。

④ **[最小距離情報]** 最小距離情報をここに更新します。

⑤ **[Auto Run Mode チェックボタン]** 本ボタンがチェックされている場合は、①で都市配置をドラッグし、ドロップした直後に自動的に経路探索を開始します。チェックされていない場合は、②の Start ボタンを押下するまで経路探索を開始しません。

⑥ **[設定項目]** 地図背景画像などを選択できます。

⑦ **[試行ごとの距離情報グラフ]** 全探索試行数(図の場合は 250)に対し、部分試行数(図の場合は 25)に区切り、部分試行の単位で①や④の情報を更新します。このグラフでは全ての試行について全て得られた距離結果を順次表示します。

停止する場合は、run.py を実行したコンソールを開き「Ctrl+C」を押下してください。

7.3.6 解説

SBM TSP ソルバの実装

GUI 版で使用する SBM TSP ソルバを、**tsp/main.cpp** で実装しています。ここでは、主となる 2 つの関数について紹介します。

$examples/tsp/main.cpp

```
213 /**
214  * Setup SBM TSP solver with the problem defined by fn_tsp(TYPE2 TSP file)
215  */
216 extern "C" void sbm_tsp_setup(const char *fn_tsp, int Ncycle, float c, float
dt,
217     float alph, float beta){
```

```
218
219     load_tsp(fn_tsp);
220     formulate_tsp2qubo(distmap_norm, Q);
221     qubo2ising(Q, J, h);
222
223     sbm_set_random_seed(18);
224
225     if(fjob != NULL) sbm_release_job(fjob);
226
227     fh = sbm_setup_fpga(E_SBM_ALGO_B, N, E_SBM_DTYPE_FP32);
228     fjob = sbm_create_job(fh, N, J, h, Ncycle, c, dt, alph, beta);
229 }
230
231 /**
232  * Run SBM TSP solver for Ntimes and update information for the best
233  */
234 extern "C" int sbm_solve_tsp(int Ntime, float *dlist, int *route){
235
236     float dmin = INFINITY;
237     int best_id = -1;
238
239     for(int ntime=0; ntime<Ntime; ntime++){
240         sbm_run_sync(fjob, s);
241         spin2bin(s, b);
242
243         float d = calc_distance(b);
244
245         if(d > 0 && d < dmin){
246             dmin = d;
247             best_id = ntime;
248             set_route_from_b(b, route);
249         }
250         dlist[ntime] = d;
251     }
252     return best_id;
253 }
```

MOT(7.2.6)では、リアルタイム性重視のため 1 つの問題に対し SBM を 1 度だけ起動する前提で組合せ最適化問題ソルバを設計していました。対して、TSP ではユースケースとして 1 つの問題に対し複数回の試行が行われることを想定しています。これを効率的に実装するため、アプリケーションが利用する SBM-TSP ソルバのインタフェースとしては、TSP では問題が決まったときに 1 度だけ呼ぶ **sbm_tsp_setup()関数**と、決まった問題に対し複数回の試行を行いその中での最良結果を返す **sbm_solve_tsp()関数**、の 2 者を定義しています。

sbm_tsp_setup()関数[L216-L229]**:** TSP 問題定義ファイル(各都市間距離リスト)のパスと SBM パラメータを入力条件とし、イジング問題定式化から SBM ジョブ(**fjob**)の生成までを実施します。TSP 問題の読み込みを **load_tsp()**関数[L219]、QUBO 形式への定式化を

formulate_tsp2qubo()関数[L220]、QUBO 形式からイジング形式への変換を**qubo2ising()**関数[L221]で行っています。これらの関数も main.cpp 内で実装されているので、詳細についてはソースコードを参照ください。

sbm_solve_tsp()関数[L234-L253]: 試行回数(Ntime)を入力とし、Ntime 回 SBM を実行[L239-L240]します。各実行結果(スピンベクトル:s)に対し、ルートの距離を計算[L243]し、制約条件を満たす場合(d > 0)には、最小距離情報の更新と最小ルートの算出を行います[L245-L249]。Ntime 回の試行のうち、分布情報として全ての試行結果(距離)を dlist に格納[L250]し、最小距離を記録した試行番号を返り値、そのルート情報を route に格納[L248]します。

CLI 版では、上記相当のソルバ実装を直接 main 関数(main.cpp)内で行っています。

SBM TSP ソルバの呼び出し

GUI 版では、アプリケーションから直接呼び出される Python 向け SBM-TSP ソルバを、**gsbm_tsp_solver()**関数として、**gui_server/py/tsp_solver.py** で実装しています。

$examples/gui_server/py/tsp_solver.py

```
24 #---------------------------------------------------------------------
25 # SBM TSP solver for python
26 #
27 # Parameters
28 #    position_file: Path of a file including positions
29 #    Ntrial : Number of  trials
30 #    Nresp  : Number of responses by this generator
31 #
32 # Returns
33 #    dists: disctances in this response
34 #    best_info :
35 #        best_info[0]: id for the best distance in this response
36 #        best_info[1]: current best route
37 #        best_info[2]: current best distance
38 #    dist_ave : current average of distances
39 #---------------------------------------------------------------------
40 def gsbm_tsp_solver(position_file, Ntrial, Nresp):
41
42     # Setup SBM tsp solver for the problem defined in 'position_file'
43     lib.sbm_tsp_setup(position_file.encode("ascii"), 1000, 0.5, 0.1, 0.5, 2.0)
44
45     Ntime = int(np.ceil(Ntrial / Nresp))
46     tmp_dists = np.zeros(Ntime, dtype=np.float32)
47     route = np.zeros(42, dtype=np.int32)
48
49     dist_total = 0.
```

```
50      valid_total = 0
51      best_dist = float('inf') # current best distance
52      best_id = None           # best id in the response
53      best_route = None        # current best route
54      all_dist = []
55
56      for i in range(Nresp):
57          # Execute SBM tsp solver on c-library
58          tmp_best_id = lib.sbm_solve_tsp(Ntime, tmp_dists, route)
59          dists = list(np.where(tmp_dists< 0, np.nan, tmp_dists))
60
61          # Update best result(best_*) if the best is in latest execution
62          if dists[tmp_best_id] < best_dist:
63              best_dist = dists[tmp_best_id]
64              best_route = list(np.copy(route))
65              best_id = tmp_best_id
66
67          best_info = [best_id, best_route, best_dist]
68          all_dist.extend(dists)
69          dist_ave = np.nanmean(all_dist)
70
71          yield dists, best_info, dist_ave
```

本関数は、アプリケーション特性に合わせ generator として実装されています。前節で定義した SBM-TSP ソルバは、 MOT(7.2.6)同様にダイナミックリンクライブラリを介してロードされ、それぞれ [L43](**sbm_tsp_setup**)、[L58](**sbm_solve_tsp**)で使用されています。

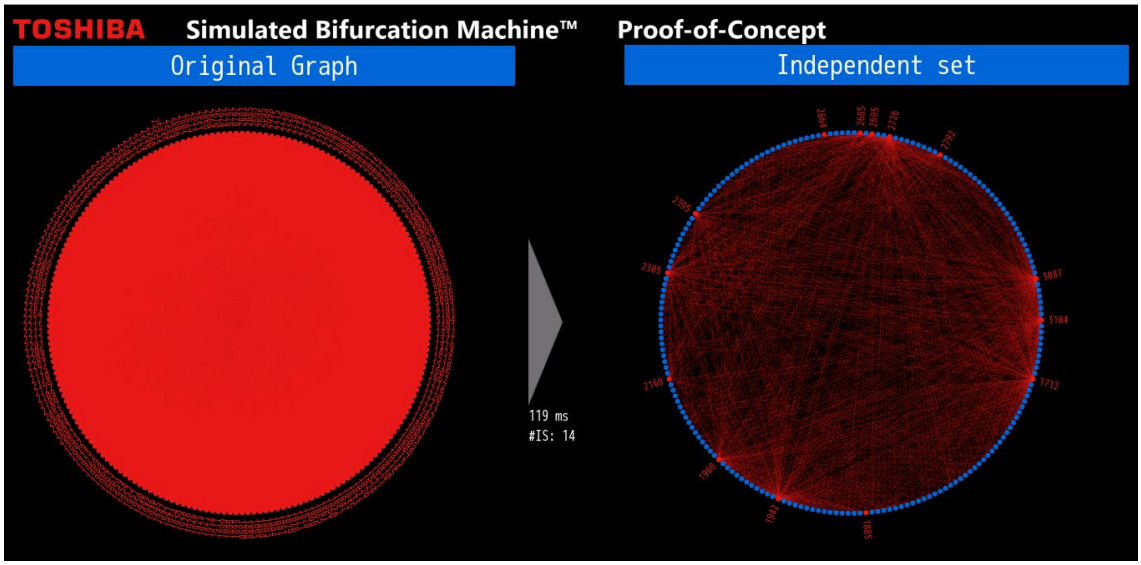
本チュートリアルではこれ以上細部に踏み込みませんが、この **gsbm_tsp_solver()**関数は、アプリケーションの一部として **gui_server/py/tsp_solver_controller.py** の中で使用されています。

7.4　最大独立集合問題ソルバ

7.4.1　概要

MIS (Maximum Independent Set: 最大独立集合問題)とは、グラフ G=（V、E）において互いに隣接しない頂点の集合（Independent Set、以降 IS と呼ぶ）のうち、その頂点数が最大の集合（Maximum Independent Set）を求める組み合わせ最適化問題です。ここでは、SBM による MIS ソルバ実装例を示した上で、疑似株式データを使って Market Graph(7.4.2)の解析を行うプログラム例(図 11)を紹介します。

図 11：MIS GUI 版画面イメージ



7.4.2　Market Graph

MIS の応用先の一つに株式ポートフォリオ最適化があげられます。本アプリケーションでは、Butenko[5]が提案する Market Graph で株式市場を表現し、この Market Graph における MIS 問題を解く例を示します。Market Graph は以下のように構築されます。

[5] S.Butenko, Maximum independent set and related problems, with applications, Doctor Dissertation, 2003.

・ Graph の頂点に各株式を対応付ける

・ 2 つの頂点間は、対応する株式のペアの(特定期間で計算された標本)相関係数が閾値θより大きければエッジで接続される

Market Graph の MIS 問題から得られる解を利用することで、多様化されたポートフォリオを構築できることが期待されます。また、Market Graph からより大きなサイズの IS を見つけることにより、多様化ポートフォリオ構築のためにより有益な手段が得られる可能性も期待できます。

本アプリケーションは、あらかじめ用意したダミーの株価データ (有効株式銘柄数:1329、 全レコード数:1440)を用い、その時系列に沿って複数個生成した Market Graph を入力とします。生成される Market Graph 数は"全レコード数 / サンプル期間(簡単のためレコード数で指定)"となります。これらの Market Graph を順次 MIS 問題として解き、CLI 版ではその結果概要をコンソールにテキスト出力、GUI 版ではグラフ表示(一定時間ごとに切り替え表示)します。

7.4.3 ファイル構成

SBM MIS ソルバおよび CLI 版に関連するファイル($examples 基点の相対パス表記)を列挙します。

ディレクトリ	ファイル名	内容
mis	requirements.txt	Python モジュールリスト
mis	mgraph.py	Market Graph 生成プログラム
mis	check_is.py	独立集合チェックプログラム
mis	mis_solver.py	各種 MIS ソルバの Python wrapper
mis	solve_mgraph.py	アプリケーションメイン Python スクリプト
mis	dummy_stocks.csv	ダミーの株価データ
mis/libsbmmis	Makefile	libsbmmis.cpp から共有ライブラリ libsbmmis.so を生成するための Makefile
mis/libsbmmis	libsbmmis.cpp	SBM による MIS ソルバの C++実装

GUI 版の関連ファイルは多数であるため、主要なものの概要のみを下記に示します。必要に応じ gui_server 以下のファイルを参照ください。

ディレクトリ	ファイル名	内容
gui_server	run.py	GUI サーバー実行スクリプト
gui_server	requirements.txt	Python モジュールリスト

gui_server/py	*.py	GUI 版 TSP・MIS 関連 Python ファイル
gui_server/static	*.js	GUI 版 TSP・MIS 関連 JavaScript ファイル

7.4.4　実行環境の準備（CLI 版・GUI 版共通）

Python パッケージのインストール

本デザイン実行に必要な Python パッケージをインストールします。

```
$ cd $examples/mis
$ pip3 install -r requirements.txt --user
```

プログラムのビルド

C/C++で実装された CLI 版および GUI 版向けの SBM-MIS ソルバをビルドします。

```
$ cd $examples/mis/libsbmmis
$ make
```

これにより、mis/libsbmmis 以下に **libsbmmis.so** (CUI/GUI 向けダイナミックリンクライブラリ)が生成されます。

7.4.5　実行環境の準備（GUI 版）

TSP と MIS で GUI 環境は共通です。既に TSP の GUI 環境を準備している場合には、MIS 向けに新たに環境を準備する必要はありません。そうでない場合は、7.3.4 を参照し、GUI 版実行環境のセットアップを実施ください。

7.4.6　実行方法

CLI 版

各種 MIS ソルバ比較のため、**mis_solver.py** には **SBM** による MIS ソルバの他、近似解ソルバである **NetworkX**、厳密解ソルバである **OR-tools** を同じインタフェースで呼び出せるよう用意しています。また、これらの MIS ソルバを利用するアプリケーション例として、**solve_mgraph.py** を用意しています。この **solve_mgraph.py** では、同梱するダミー株式データ(dummy_stocks.csv)か

ら任意のノード数(-N)・サンプル期間(-d)で Market Graph を作成します。作成された Market Graph の時系列データを対象とし、指定するソルバ(-s)を用いて MIS 問題を解きます。結果として得た独立集合の数と演算に要した時間をコンソールに表示します。ソルバが SBM の場合、SBM 動作条件は下記のとおりです。

- SBM アルゴリズム: E_SBM_ALGO_B
- J 行列精度: E_SBM_DTYPE_FP32
- [step, c, dt, alph, beta]: 1000, 0.4, 0.3, 1.0, 1.0

ソルバが SBM の場合は、上記設定で 10 回実行し、制約条件を満たした試行の数と独立集合数の平均値も合わせて表示します。

CLI 版 MIS ソルバは、$examples/mis ディレクトリ下で実行します。

```
$ cd $examples/mis
```

以下は、[ソルバ, ノード数, サンプル期間]=[**SBM**, 400, 480]の場合の実行例です。

```
$ python3 solve_mgraph.py -s sbm -d 480 -N 400
INFO : Time of fpga_setup: 0.127222799 s
[      120 ms] 10/10 #is(ave,best)=39.8, 40
[      120 ms] 10/10 #is(ave,best)=134.2, 135
[      120 ms] 10/10 #is(ave,best)=15.2, 16
```

以下は、近似解ソルバ **NetworkX** を用いた場合**(注)**の実行例(ソルバ以外は同条件)です。

```
$ python3 solve_mgraph.py -s nx -d 480 -N 400
INFO : Time of fpga_setup: 0.127416157 s
[    6656 ms] #is=35
[    9785 ms] #is=122
[    3384 ms] #is=13
```

以下は、厳密解ソルバ **OR-tools** を用いた場合**(注)**の実行例(ソルバ以外は同条件)です。

```
$ python3 solve_mgraph.py -s ort -d 480 -N 400
INFO : Time of fpga_setup: 0.125679123 s
[ 156078 ms] #is=40
[   12182 ms] #is=135
[ 995966 ms] #is=17
```

solve_mgraph.py の初回実行時には、dummy_stocks.csv から Pandas データフレーム化し、さらに前処理を実施したのちに pickle 形式(dummy_stocks.**pkl**)で保存する処理が行われるため、ソルバ実行前のデータ処理に多くの時間を要します。2 回目以降の実行時には、前処理済みの pkl ファイルを直接ロードするためデータ処理は高速です。

(注) ソルバに **NetworkX** または **OR-tools** を選んだ場合、MIS ソルバ動作に SBM は使用しません(SBM で NetworkX や OR-tools を動作させる、という意味ではありません)。

GUI 版

GUI 版は、GUI としてブラウザを使用します。まず、下記のように run.py を実行し、ローカルサーバを起動します。

```
$ cd $examples/gui_server
$ python3 run.py
```

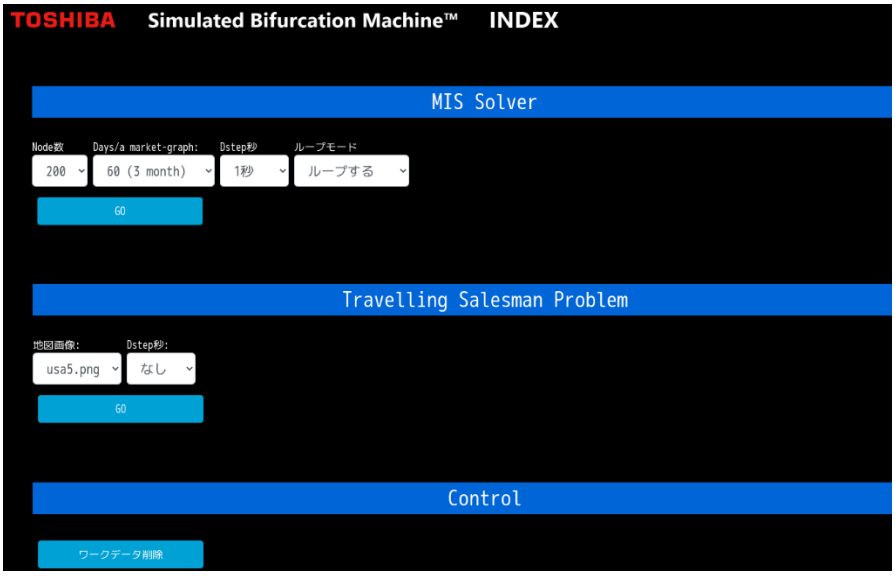
正常に起動されると、下記のようにコンソールに URL が表示されます。

上記の場合、ブラウザで「**http://127.0.0.1:5042**」にアクセスしてください。正常動作している場合

```
INFO:     Started server process [6933]
INFO:     Uvicorn running on http://127.0.0.1:5042 (Press CTRL+C to quit)
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

は、図 12 のインデックスページが表示されます。

図 12：TSP/MIS GUI 版アプリインデックスページ



インデックスページ中の MIS Solver グループ内にある各種パラメータ項目を確認(必要あれば再設定**(注)**)し、GO ボタンを押下してください。押下後、必要なパラメータが設定され図 11 の画面に遷移します。図 11 の画面では、左側に元の Market Graph、右側に IS として抽出されたアセットを強調表示したグラフを表示します。ユーザーから操作可能な UI はなく、一定時間ごとに自動的に表示が切り替わります。各種パラメータ(ノード数や、サンプル期間など)を変更する場合には、図 12 のインデックスページに戻って行います。

(注) ノード数が大きい場合(800 など)、ブラウザへのグラフ描画処理の負荷が大きくなるため、画面更新速度の低下やページの応答がなくなる可能性があります。

停止する場合は、run.py を実行したコンソールを開き「Ctrl+C」を押下してください。

7.4.7 解説

SBM MIS ソルバの実装 (C 言語)

SBM による MIS ソルバを C 言語（**$examples/mis/libsbmmis/libsbmmis.cpp**）で実装します。

MIS 問題を QUBO 形式に定式化

まず、MIS の(QUBO に対する)定式化に関する 2 つの関数の実装について紹介します。

$examples/mis/libsbmmis/libsbmmis.cpp

```
20 /**
21  * Formulate MIS defined by G w/ A,B to QUBO format
22  */
23 void formulate_mis2qubo(int N, float *G, float A, float B, float *Q){
24
25     memset(Q, 0, N*N*sizeof(float));
26
27     for(int i=0; i<N; i++){
28         for(int j=i+1; j<N; j++){
29             if(G[i*N+j] != 0){
30                 Q[i*N+j] = Q[j*N+i] = A;
31             }
32         }
33     }
34
35     for(int i=0; i<N; i++){
```

```
 36         Q[i*N+i] -= B;
 37     }
 38 }
-------------- 中略 --------------
 57 /**
 58  * Check if 'ans'(related to 'G') is Independent Set of not
 59  */
 60 int check_is(int N, float *G, int *ans){
 61     int cnt = 0;
 62     for(int i=0; i<N; i++){
 63         if(ans[i] > 0){
 64             cand_idx[cnt] = i;
 65             cnt++;
 66         }
 67     }
 68
 69     for(int i=0; i<cnt; i++){
 70         for(int j=i+1; j<cnt; j++){
 71             if(G[cand_idx[i]*N + cand_idx[j]] > 0)
 72                 return 0;
 73         }
 74     }
 75     return cnt;
 76 }
```

formulate_mis2qubo()関数[L23-L38]: ノード数 **N** の(N, N)グラフ **G** 入力とし、QUBO 定式化された(N, N)行列 **Q** を出力とします。MIS から QUBO 形式への定式化方法は、参考文献 *"Ising formulations of many NP problems"*[6]の*"4.2. Set Packing"*における定義式に準じます。ここでは、それぞれの係数を A, B とし、$\mathrm{H} = \mathrm{H_A} + \mathrm{H_B}$における制約項($\mathrm{H_A} = A\sum_{ij\in E} x_i x_j$)を[L27-L33]、コスト項($\mathrm{H_B} = -B\sum_i x_i$)を[L35-L37]で計算しています。

check_is()関数[L60-L76]: ノード数 **N** の(N, N)グラフ **G** に対し、独立集合候補列 **ans** (**ans**[i]=1 のときノード i が独立集合の要素であることを示す)が独立集合であるか否かを判断します。**ans** が **G** の独立集合でない場合は 0 を、独立集合である場合はその要素数を返します。[L62-L67]では、**ans**[i]が 1 である数をカウント(独立集合要素数候補)しながら、一時変数 cand_idx[]に **ans**[i]が 1 であるインデックス(i)を追記します。[L69-L74]では、グラフ **G** と

[6] Andrew Lucas, Ising formulations of many NP problem
(https://www.frontiersin.org/articles/10.3389/fphy.2014.00005/full)

cand_idx[]を照らし合わせながら、制約項($H_A = A\sum_{ij\in E} x_i x_j$)の全ての i,j の項について 0 であることを確認します。

Python スクリプトから呼び出される MIS ソルバ関数(C 言語)の実装

前節の **formulate_mis2qubo()**関数、**check_is()**関数と、QUBO 形式からイジング形式への変換を行う **qubo2ising()**関数、および SBM API 関数を使って、Python スクリプトとのインタフェースとなる MIS ソルバ関数(**sbm_mis_solver()**)を実装します。

$examples/mis/libsbmmis/libsbmmis.cpp

```
 85 /**
 86  * SBM MIS solver : Problem input is G (Original graph)
 87  */
 88 extern "C" int sbm_mis_solver(int N, float *G, float A, float B, int Ntime,
 89     int Ncycle, float c, float dt, float alph, float beta,
 90     int *ans, int *p_best_isnum, int *p_num_ok, float *p_ave_isnum){
-------------- 中略 --------------
102     // Formulate MIS(G)w/(A,B) -> QUBO(Q) -> Ising(J,h)
103     formulate_mis2qubo(N, G, A, B, Q);
104
105     qubo2ising(N, Q, J, h);
106
107     // FPGA initial setup
108     sbm_set_random_seed(1);
109     H_SBM_FPGA fh = sbm_setup_fpga(E_SBM_ALGO_B, 2000, E_SBM_DTYPE_FP32);
110     H_SBM_JOB fjob = sbm_create_job(fh, N, J, h, Ncycle, c, dt, alph, beta);
-------------- 中略 --------------
117     // Calculate and Update the result by SBM for the number of Ntime loop
118     for(int ntime=0; ntime<Ntime; ntime++){
119         sbm_run_sync(fjob, g_s);
120
121         int isnum = check_is(N, G, g_s);
122         if(isnum > 0){
123             ok_cnt++;
124             ave_isnum += isnum;
125             if(isnum > best_isnum){
126                 best_isnum = isnum;
127                 for(int i=0; i<N; i++) ans[i] = g_s[i];
128             }
129         }
130     }
-------------- 中略 --------------
141     // Finalize
142     sbm_release_job(fjob);
-------------- 中略 --------------
148 }
```

[L103]: グラフ **G** を QUBO 形式 **Q** に定式化します。

[L105]: QUBO 形式 **Q** をイジング形式(**J**, **h**)に変換します。

[L109-110]: SBM-API を呼び出し、FPGA のセットアップおよびジョブのインスタンス化を行います。同一プロセス内で既に FPGA にロードされている回路インスタンスと同じ回路インスタンスを指定して 2 回目以降の **sbm_setup_fpga()**を実施する場合、FPGA への(無意味な)再ロードは行われません。そのため、sbm_mis_solver()の呼び出し毎に、**sbm_setup_fpga()**が呼び出されても、2 回目以降の再ロードコスト(時間)はかかりません。

[L119]: SBM でジョブを実行します(同期モード)

[L121-129]: SBM 実行により得た解(**g_s**)が、制約条件を満たしているか確認し、制約条件を満たし、かつループ内でこれまでの試行で得られた独立集合数のベスト値を更新する場合は、sbm_mis_solver()呼び出し側に返す解 **ans** に **g_s** をコピーします。

[L142]: SBM ジョブをリリースします。(ジョブ実行のために確保したメモリを解放します)

SBM MIS ソルバの実装 (Python)

C 言語で実装した **sbm_mis_solver()**関数(前節)を Python スクリプトから呼び出せるようにします。ここでは、Python スクリプト側で定義する関数(ラッパ関数)を **solve_by_sbm()**としています。

$examples/mis/mis_solver.py

```
 11 import numpy.ctypeslib as npct
 12 from ctypes import *
 13
 14 # C で実装された SBM MIS ソルバを Python で使用するための準備
 15 libdir = os.path.dirname(os.path.abspath(__file__)) + '/libsbmmis'
 16 lib = npct.load_library('libsbmmis.so', libdir)
 17 c_af1 = npct.ndpointer(dtype=np.float32, ndim=1, flags='C_CONTIGUOUS')
 18 c_ai1 = npct.ndpointer(dtype=np.int32, ndim=1, flags='C_CONTIGUOUS')
-------------- 中略 --------------
 23 lib.sbm_mis_solver.argtypes = [c_int, c_ai1, c_float, c_float, c_int, c_int,
 24     c_float, c_float, c_float, c_float, c_ai1, c_ai1, c_ai1, c_af1]
 25 lib.sbm_mis_solver.restype = c_int
-------------- 中略 --------------
 62 def solve_by_sbm(G, A, B, Ntimes, Ncycle, c, dt, alph, beta):
 63     N = G.shape[0]
```

```
64
65      best_isnum = np.zeros(1, dtype=np.int32)
66      ok_cnt = np.zeros(1, dtype=np.int32)
67      ave_isnum = np.zeros(1, dtype=np.float32)
68      ans = np.zeros(N, dtype=np.int32)
69
70      lib.sbm_mis_solver(N, G.flatten(), c_float(A), c_float(B),
71          Ntimes, Ncycle, c_float(c), c_float(dt), c_float(alph), c_float(beta),
72          ans, best_isnum, ok_cnt, ave_isnum)
73
74      return np.where(ans < 0, 0, 1), best_isnum[0], ok_cnt[0], ave_isnum[0]
```

ラッパ実装の考え方については、7.2.6 “B. A のソルバを Python から呼び出す手順”と同様です。

SBM MIS ソルバの呼び出し

前節で定義した Python 版 SBM-MIS ソルバ(**solve_by_sbm()**)の使用例(使用箇所)について説明します。

CLI 版

```
$examples/mis/solve_mgraph.py
15 from mis_solver import solve_by_sbm
-------------- 中略 --------------
47      df_ln_full = load_stockfile('./dummy_stocks')
48      df_ln = df_ln_full.iloc[:, :args.N]
49
50      for i in range(0, len(df_ln), dur):
51
52          # generate market graph
53          G = market_graph(df_ln, theta=theta, ise=[i, i+dur]).astype(np.int32)
54
55          if args.solver == 'sbm':
56              # solve by SBM MIS solver
57              time0 = time.time()
58              answer, isnum, okcnt, ave_isnum = solve_by_sbm(G, A, B, Ntimes,
59                  Ncycle, c, dt, alph, beta)
```

株式データ(dummy_stocks.csv)の前処理済みデータフレームをロード[L47]し、スクリプトへの引数指定するノード数 **N** を切り出します[L48](→ **df_ln**)。**df_ln** の先頭から順次 **dur** 期間(row 数)分のデータを対象に Market Graph(市場グラフ)**G** を作成します[L53]。SBM-MIS ソルバ(**solve_by_sbm()**)を動作させ、G に対する結果(MIS 行列(**answer**)、ベスト IS 数、有効解を得た試行数、IS 数平均)を取得します[L58]。

GUI 版

$examples/gui_server/py/mis_data.py

```
 56             # generate market graph
 57             ise = [index*self.mgraphDuration, (index+1)*self.mgraphDuration]
 58             G = mgraph.market_graph(self.df_ln, 0.3, ise=ise).astype(np.int32)
 59
 60             # solve MIS by SBM
 61             time0 = time.time()
 62             ans, isnum, _, _ = solve_by_sbm(G, A, B, Ntimes, Ncycle, c, dt,
 63                 alph, beta)
 64             calc_time = (time.time() - time0)*1000
```

使用方法(呼び出し方法)は CLI 版と概ね同様です(詳細説明は割愛します)。

第8章　Appendix

8.1　Disclaimer

THE SOFTWARE AND SERVICES ARE PROVIDED "AS IS." YOU AND YOUR USERS ACCEPT AND ASSUME THE ENTIRE RISK AS TO THE QUALITY, PERFORMANCE AND RESULTS OF ACCESS OR USE OF THE SOFTWARE AND SERVICES. TOSHIBA DIGITAL SOLUTIONS CORPORATION (TDSL) AND TDSL AFFILIATES AND LICENSORS MAKE NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE REGARDING THE SOFTWARE AND SERVICES, INCLUDING ANY WARRANTY THAT THE SOFTWARE AND SERVICES WILL BE UNINTERRUPTED, ERROR FREE OR FREE OF HARMFUL COMPONENTS, OR THAT ANY CONTENT IN YOUR COMPUTER, INCLUDING YOUR CONTENT AND DATA, WILL BE SECURE OR NOT OTHERWISE LOST OR DAMAGED. EXCEPT TO THE EXTENT PROHIBITED BY LAW, TDSL AND TDSL AFFILIATES AND LICENSORS DISCLAIM ALL WARRANTIES, INCLUDING ANY IMPLIED WARRANTIES OF PERFORMANCE, MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR QUIET ENJOYMENT, AND ANY WARRANTIES ARISING OUT OF ANY COURSE OF DEALING OR USAGE OF TRADE.

8.2 License

Copyright (C) 2013-2016 Altera Corporation, San Jose, California, USA. All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

This agreement shall be governed in all respects by the laws of the State of California and by the laws of the United States of America.

A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using init_genrand(seed) or init_by_array(init_key, key_length).

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

LICENSE. Licensee has a license under Intel's copyrights to reproduce Intel's Software only in its unmodified and binary form, (with the accompanying documentation, the "Software") for Licensee's personal use only, and not commercial use, in connection with Intel-based products for which the Software has been provided, subject to the following conditions:

(a) Licensee may not disclose, distribute or transfer any part of the Software, and Licensee agree to prevent unauthorized copying of the Software.

(b) Licensee may not reverse engineer, decompile, or disassemble the Software.

(c) Licensee may not sublicense the Software.

(d) The Software may contain the software and other intellectual property of third party suppliers, some of which may be identified in, and licensed in accordance with, an enclosed license.txt file or other text or file.

(e) Intel has no obligation to provide any support, technical assistance or updates for the Software.

OWNERSHIP OF SOFTWARE AND COPYRIGHTS. Title to all copies of the Software remains with Intel or its licensors or suppliers. The Software is copyrighted and protected by the laws of the United States and other countries, and international treaty provisions. Licensee may not remove any copyright notices from the Software. Except as otherwise expressly provided above, Intel grants no express or implied right under Intel patents, copyrights, trademarks, or other intellectual property rights. Transfer of the license terminates Licensee's right to use the Software.

DISCLAIMER OF WARRANTY. The Software is provided "AS IS" without warranty of any kind, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.

LIMITATION OF LIABILITY. NEITHER INTEL NOR ITS LICENSORS OR SUPPLIERS WILL BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF USE, INTERRUPTION OF BUSINESS, OR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY KIND WHETHER UNDER THIS AGREEMENT OR OTHERWISE, EVEN IF INTEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

LICENSE TO USE COMMENTS AND SUGGESTIONS. This Agreement does NOT obligate Licensee to provide Intel with comments or suggestions regarding the Software. However, if Licensee provides Intel with comments or suggestions for the modification, correction, improvement or enhancement of (a) the Software or (b) Intel products or processes that work with the Software, Licensee grants to Intel a non-exclusive, worldwide, perpetual, irrevocable, transferable, royalty-free license, with the right to sublicense, under Licensee's intellectual property rights, to incorporate or otherwise utilize those comments and suggestions.

TERMINATION OF THIS LICENSE. Intel or the sublicensor may terminate this license at any time if Licensee is in breach of any of its terms or conditions. Upon termination, Licensee will immediately destroy or return to Intel all copies of the Software.

THIRD PARTY BENEFICIARY. Intel is an intended beneficiary of the End User License Agreement and has the right to enforce all of its terms.

U.S. GOVERNMENT RESTRICTED RIGHTS. The Software is a commercial item (as defined in 48 C.F.R. 2.101) consisting of commercial computer software and commercial computer software documentation (as those terms are used in 48 C.F.R. 12.212), consistent with 48 C.F.R. 12.212 and 48 C.F.R 227.7202-1 through 227.7202-4. Licensee will not provide the Software to the U.S. Government. Contractor or Manufacturer is Intel Corporation, 2200 Mission College Blvd., Santa Clara, CA 95054.

EXPORT LAWS. Licensee agrees that neither Licensee nor Licensee's subsidiaries will export/re-export the Software, directly or indirectly, to any country for which the U.S. Department of Commerce or any other agency or department of the U.S. Government or the foreign government from where it is shipping requires an export license, or other governmental approval, without first obtaining any such required license or approval. In the event the Software is exported from the U.S.A. or re-exported from a foreign destination by Licensee, Licensee will ensure that the distribution and export/re-export or import of the Software complies with all laws, regulations, orders, or other restrictions of the U.S. Export Administration Regulations and the appropriate foreign government.

APPLICABLE LAWS. This Agreement and any dispute arising out of or relating to it will be governed by the laws of the U.S.A. and Delaware, without regard to conflict of laws principles. The Parties to this Agreement exclude the application of the United Nations Convention on Contracts for the International Sale of Goods (1980). The state and federal courts sitting in Delaware, U.S.A. will have exclusive jurisdiction over any dispute arising out of or relating to this Agreement. The Parties consent to personal jurisdiction and venue in those courts. A Party that obtains a judgment against the other Party in the courts identified in this section may enforce that judgment in any court that has jurisdiction over the Parties.

Licensee's specific rights may vary from country to country.

8.3 診断プログラムについて

診断プログラムで使われる問題は以下の通りです。

- 最大カット問題ベンチマークセット Gset（G1 - G65）（*）

インスタンス	ノード	エッジ	重み	タイプ
G1	800	19176	+1	random
G2	800	19176	+1	random
G3	800	19176	+1	random
G4	800	19176	+1	random
G5	800	19176	+1	random
G6	800	19176	+1,-1	random
G7	800	19176	+1,-1	random
G8	800	19176	+1,-1	random

G9	800	19176	+1,-1	random
G10	800	19176	+1,-1	random
G11	800	1600	+1,-1	toroidal
G12	800	1600	+1,-1	toroidal
G13	800	1600	+1,-1	toroidal
G14	800	4694	+1	planar
G15	800	4661	+1	planar
G16	800	4672	+1	planar
G17	800	4667	+1	planar
G18	800	4694	+1,-1	planar
G19	800	4661	+1,-1	planar
G20	800	4672	+1,-1	planar
G21	800	4667	+1,-1	planar
G22	2000	19990	+1	random
G23	2000	19990	+1	random
G24	2000	19990	+1	random
G25	2000	19990	+1	random
G26	2000	19990	+1	random
G27	2000	19990	+1,-1	random
G28	2000	19990	+1,-1	random
G29	2000	19990	+1,-1	random
G30	2000	19990	+1,-1	random
G31	2000	19990	+1,-1	random
G32	2000	4000	+1,-1	toroidal
G33	2000	4000	+1,-1	toroidal
G34	2000	4000	+1,-1	toroidal
G35	2000	11778	+1	planar
G36	2000	11766	+1	planar
G37	2000	11785	+1	planar
G38	2000	11779	+1	planar
G39	2000	11778	+1,-1	planar
G40	2000	11766	+1,-1	planar
G41	2000	11785	+1,-1	planar
G42	2000	11779	+1,-1	planar
G43	1000	9990	+1	random

G44	1000	9990	+1	random
G45	1000	9990	+1	random
G46	1000	9990	+1	random
G47	1000	9990	+1	random
G48	3000	6000	+1,-1	toroidal
G49	3000	6000	+1,-1	toroidal
G50	3000	6000	+1,-1	toroidal
G51	1000	5909	+1	planar
G52	1000	5916	+1	planar
G53	1000	5914	+1	planar
G54	1000	5916	+1	planar
G55	5000	12498	+1	random
G56	5000	12498	+1,-1	random
G57	5000	10000	+1,-1	toroidal
G58	5000	29570	+1	planar
G59	5000	29570	+1,-1	planar
G60	7000	17148	+1	random
G61	7000	17148	+1,-1	random
G62	7000	14000	+1,-1	toroidal
G63	7000	41459	+1	planar
G64	7000	41459	+1,-1	planar
G65	8000	16000	+1,-1	toroidal

- Sherrington–Kirkpatrick 模型（SK 模型）K2000 問題
- Sherrington–Kirkpatrick 模型（SK 模型）K1024 問題（係数付き）
- 巡回セールスマン問題 Dantzig42

*1：スピン数 8192 まで対応可能です。

診断プログラムの実行ログ例を以下に示します。

```
--- G,K,M check (start) ---
make: `all' に対して行うべき事はありません.
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac_s10_dc : Intel PAC Platform (pac_f100000)
```

```
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048FP32 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048FP32 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 3.080952558 s
K2000 : cut(average) = 33154.64 / 33337 ( 99.45% ),
FPGA kernel time (core time) = 0.003869069 ( 0.001720463 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048FP32 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048FP32 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 3.042665590 s
K2000 : cut(average) = 33171.72 / 33337 ( 99.50% ),
FPGA kernel time (core time) = 0.010845922 ( 0.008698039 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048INT2 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048INT2 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 3.058274936 s
K2000 : cut(average) = 33182.28 / 33337 ( 99.54% ),
FPGA kernel time (core time) = 0.001774803 ( 0.000719745 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C8192INT2 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING C8192INT2 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 3.379006480 s
K2000 : cut(average) = 33182.28 / 33337 ( 99.54% ),
FPGA kernel time (core time) = 0.024711113 ( 0.007291667 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048INT1 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048INT1 1 0.aocx
DEBUG : num_devices 1
```

```
INFO : Time of fpga setup: 3.275159169 s
K2000 : cut(average) = 33174.73 / 33337 ( 99.51% ),
FPGA kernel time (core time) = 0.000759409 ( 0.000273540 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048FP32 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048FP32 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 3.002021749 s
M1024 : cut(average) = 238550448.00 / 240502032 ( 99.19% ),
FPGA kernel time (core time) = 0.002948908 ( 0.000774208 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048FP32 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048FP32 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 2.969874519 s
M1024 : cut(average) = 238441696.00 / 240502032 ( 99.14% ),
FPGA kernel time (core time) = 0.005636149 ( 0.003479216 ) s ->
PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048FP32 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING B2048FP32 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 3.002588948 s
  G1 : cut(average) = 11547.42 / 11624 ( 99.34% ), FPGA kernel time
(core time) = 0.002420594 ( 0.000258069 ) s -> PASSED
  G2 : cut(average) = 11545.96 / 11620 ( 99.36% ), FPGA kernel time
(core time) = 0.002421316 ( 0.000258069 ) s -> PASSED
  G3 : cut(average) = 11549.38 / 11622 ( 99.38% ), FPGA kernel time
(core time) = 0.002421035 ( 0.000258069 ) s -> PASSED
  G4 : cut(average) = 11575.79 / 11646 ( 99.40% ), FPGA kernel time
(core time) = 0.002421106 ( 0.000258069 ) s -> PASSED
  G5 : cut(average) = 11563.54 / 11631 ( 99.42% ), FPGA kernel time
(core time) = 0.002420041 ( 0.000258069 ) s -> PASSED
  G6 : cut(average) =  2171.46 /  2178 ( 99.70% ), FPGA kernel time
(core time) = 0.003171654 ( 0.001032278 ) s -> PASSED
  G7 : cut(average) =  1986.20 /  2006 ( 99.01% ), FPGA kernel time
(core time) = 0.002947048 ( 0.000774208 ) s -> PASSED
  G8 : cut(average) =  1991.01 /  2005 ( 99.30% ), FPGA kernel time
(core_time) = 0.003171125 ( 0.001032278 ) s -> PASSED
```

```
  G9 : cut(average) =  2030.65 /  2054 ( 98.86% ), FPGA kernel time
(core time) = 0.002687779 ( 0.000516139 ) s -> PASSED
 G10 : cut(average) =  1982.54 /  2000 ( 99.13% ), FPGA kernel time
(core time) = 0.003278088 ( 0.001118301 ) s -> PASSED
 G11 : cut(average) =   553.14 /   564 ( 98.07% ), FPGA kernel time
(core time) = 0.003871943 ( 0.001720463 ) s -> PASSED
 G12 : cut(average) =   546.64 /   556 ( 98.32% ), FPGA kernel time
(core time) = 0.003439448 ( 0.001290347 ) s -> PASSED
 G13 : cut(average) =   570.44 /   582 ( 98.01% ), FPGA kernel time
(core time) = 0.003868122 ( 0.001720463 ) s -> PASSED
 G14 : cut(average) =  3026.95 /  3064 ( 98.79% ), FPGA kernel time
(core time) = 0.002513942 ( 0.000344093 ) s -> PASSED
 G15 : cut(average) =  3007.54 /  3050 ( 98.61% ), FPGA kernel time
(core time) = 0.002419728 ( 0.000258069 ) s -> PASSED
 G16 : cut(average) =  3013.83 /  3052 ( 98.75% ), FPGA kernel time
(core time) = 0.002508846 ( 0.000344093 ) s -> PASSED
 G17 : cut(average) =  3008.02 /  3047 ( 98.72% ), FPGA kernel time
(core time) = 0.002515448 ( 0.000344093 ) s -> PASSED
 G18 : cut(average) =   949.35 /   992 ( 95.70% ), FPGA kernel time
(core time) = 0.002425891 ( 0.000258069 ) s -> PASSED
 G19 : cut(average) =   865.18 /   906 ( 95.49% ), FPGA kernel time
(core time) = 0.002585524 ( 0.000430116 ) s -> PASSED
 G20 : cut(average) =   898.48 /   941 ( 95.48% ), FPGA kernel time
(core time) = 0.002581689 ( 0.000430116 ) s -> PASSED
 G21 : cut(average) =   886.50 /   931 ( 95.22% ), FPGA kernel time
(core time) = 0.002581892 ( 0.000430116 ) s -> PASSED
 G22 : cut(average) = 13271.29 / 13359 ( 99.34% ), FPGA kernel time
(core time) = 0.002532592 ( 0.000387104 ) s -> PASSED
 G23 : cut(average) = 13264.37 / 13344 ( 99.40% ), FPGA kernel time
(core time) = 0.002509396 ( 0.000344093 ) s -> PASSED
 G24 : cut(average) = 13260.25 / 13337 ( 99.42% ), FPGA kernel time
(core time) = 0.002514330 ( 0.000344093 ) s -> PASSED
 G25 : cut(average) = 13261.26 / 13340 ( 99.41% ), FPGA kernel time
(core time) = 0.002516407 ( 0.000344093 ) s -> PASSED
 G26 : cut(average) = 13251.75 / 13328 ( 99.43% ), FPGA kernel time
(core time) = 0.002508634 ( 0.000344093 ) s -> PASSED
 G27 : cut(average) =  3305.15 /  3341 ( 98.93% ), FPGA kernel time
(core time) = 0.003545544 ( 0.001376371 ) s -> PASSED
 G28 : cut(average) =  3263.95 /  3298 ( 98.97% ), FPGA kernel time
(core time) = 0.003116321 ( 0.000946255 ) s -> PASSED
 G29 : cut(average) =  3362.67 /  3405 ( 98.76% ), FPGA kernel time
(core time) = 0.003544958 ( 0.001376371 ) s -> PASSED
 G30 : cut(average) =  3377.20 /  3413 ( 98.95% ), FPGA kernel time
(core time) = 0.004143825 ( 0.001978533 ) s -> PASSED
 G31 : cut(average) =  3271.62 /  3310 ( 98.84% ), FPGA kernel time
(core time) = 0.003713546 ( 0.001548417 ) s -> PASSED
 G32 : cut(average) =  1375.90 /  1410 ( 97.58% ), FPGA kernel time
(core time) = 0.003711740 ( 0.001548417 ) s -> PASSED
 G33 : cut(average) =  1359.24 /  1382 ( 98.35% ), FPGA kernel time
(core time) = 0.006442061 ( 0.004301158 ) s -> PASSED
 G34 : cut(average) =  1365.32 /  1384 ( 98.65% ), FPGA kernel time
(core_time) = 0.010756169 ( 0.008602317 ) s -> PASSED
```

```
 G35 : cut(average) =  7594.55 /  7687 ( 98.80% ), FPGA kernel time
(core time) = 0.003118309 ( 0.000946255 ) s -> PASSED
 G36 : cut(average) =  7589.97 /  7680 ( 98.83% ), FPGA kernel time
(core time) = 0.003117322 ( 0.000946255 ) s -> PASSED
 G37 : cut(average) =  7580.21 /  7691 ( 98.56% ), FPGA kernel time
(core time) = 0.002748151 ( 0.000602162 ) s -> PASSED
 G38 : cut(average) =  7590.10 /  7688 ( 98.73% ), FPGA kernel time
(core time) = 0.002583940 ( 0.000430116 ) s -> PASSED
 G39 : cut(average) =  2300.49 /  2408 ( 95.54% ), FPGA kernel time
(core time) = 0.002583501 ( 0.000430116 ) s -> PASSED
 G40 : cut(average) =  2289.49 /  2400 ( 95.40% ), FPGA kernel time
(core time) = 0.002583624 ( 0.000430116 ) s -> PASSED
 G41 : cut(average) =  2283.23 /  2405 ( 94.94% ), FPGA kernel time
(core time) = 0.002589251 ( 0.000430116 ) s -> PASSED
 G42 : cut(average) =  2372.53 /  2481 ( 95.63% ), FPGA kernel time
(core time) = 0.002587206 ( 0.000430116 ) s -> PASSED
 G43 : cut(average) =  6608.05 /  6660 ( 99.22% ), FPGA kernel time
(core time) = 0.002425087 ( 0.000258069 ) s -> PASSED
 G44 : cut(average) =  6601.52 /  6650 ( 99.27% ), FPGA kernel time
(core time) = 0.002419830 ( 0.000258069 ) s -> PASSED
 G45 : cut(average) =  6604.15 /  6654 ( 99.25% ), FPGA kernel time
(core time) = 0.002420417 ( 0.000258069 ) s -> PASSED
 G46 : cut(average) =  6600.81 /  6649 ( 99.28% ), FPGA kernel time
(core time) = 0.002419780 ( 0.000258069 ) s -> PASSED
 G47 : cut(average) =  6608.98 /  6657 ( 99.28% ), FPGA kernel time
(core time) = 0.002419508 ( 0.000258069 ) s -> PASSED
 G51 : cut(average) =  3803.49 /  3848 ( 98.84% ), FPGA kernel time
(core time) = 0.002582140 ( 0.000430116 ) s -> PASSED
 G52 : cut(average) =  3804.08 /  3851 ( 98.78% ), FPGA kernel time
(core time) = 0.002516864 ( 0.000344093 ) s -> PASSED
 G53 : cut(average) =  3805.32 /  3850 ( 98.84% ), FPGA kernel time
(core time) = 0.002740811 ( 0.000602162 ) s -> PASSED
 G54 : cut(average) =  3804.66 /  3852 ( 98.77% ), FPGA kernel time
(core time) = 0.002942275 ( 0.000774208 ) s -> PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048FP32 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048FP32 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 2.972157910 s
  G1 : cut(average) = 11541.68 / 11624 ( 99.29% ), FPGA kernel time
(core time) = 0.003472679 ( 0.001304706 ) s -> PASSED
  G2 : cut(average) = 11527.70 / 11620 ( 99.21% ), FPGA kernel time
(core time) = 0.003012797 ( 0.000869804 ) s -> PASSED
  G3 : cut(average) = 11526.31 / 11622 ( 99.18% ), FPGA kernel time
(core time) = 0.003011779 ( 0.000869804 ) s -> PASSED
  G4 : cut(average) = 11525.85 / 11646 ( 98.97% ), FPGA kernel time
(core time) = 0.002582382 ( 0.000434902 ) s -> PASSED
  G5 : cut(average) = 11516.64 / 11631 ( 99.02% ), FPGA kernel time
(core_time) = 0.002583652 ( 0.000434902 ) s -> PASSED
```

```
  G6 : cut(average) =  2169.74 /  2178 ( 99.62% ), FPGA kernel time
(core time) = 0.006495449 ( 0.004349020 ) s -> PASSED
  G7 : cut(average) =  1988.32 /  2006 ( 99.12% ), FPGA kernel time
(core time) = 0.006496447 ( 0.004349020 ) s -> PASSED
  G8 : cut(average) =  1986.10 /  2005 ( 99.06% ), FPGA kernel time
(core time) = 0.004774121 ( 0.002609412 ) s -> PASSED
  G9 : cut(average) =  2030.04 /  2054 ( 98.83% ), FPGA kernel time
(core time) = 0.004772506 ( 0.002609412 ) s -> PASSED
 G10 : cut(average) =  1979.84 /  2000 ( 98.99% ), FPGA kernel time
(core time) = 0.004774823 ( 0.002609412 ) s -> PASSED
 G11 : cut(average) =   558.90 /   564 ( 99.10% ), FPGA kernel time
(core time) = 0.010851217 ( 0.008698039 ) s -> PASSED
 G12 : cut(average) =   550.54 /   556 ( 99.02% ), FPGA kernel time
(core time) = 0.010853583 ( 0.008698039 ) s -> PASSED
 G13 : cut(average) =   574.24 /   582 ( 98.67% ), FPGA kernel time
(core time) = 0.010853823 ( 0.008698039 ) s -> PASSED
 G14 : cut(average) =  3039.46 /  3064 ( 99.20% ), FPGA kernel time
(core time) = 0.004343771 ( 0.002174510 ) s -> PASSED
 G15 : cut(average) =  3024.59 /  3050 ( 99.17% ), FPGA kernel time
(core time) = 0.004346346 ( 0.002174510 ) s -> PASSED
 G16 : cut(average) =  3027.70 /  3052 ( 99.20% ), FPGA kernel time
(core time) = 0.005207782 ( 0.003044314 ) s -> PASSED
 G17 : cut(average) =  3021.72 /  3047 ( 99.17% ), FPGA kernel time
(core time) = 0.004342722 ( 0.002174510 ) s -> PASSED
 G18 : cut(average) =   974.51 /   992 ( 98.24% ), FPGA kernel time
(core time) = 0.005637702 ( 0.003479216 ) s -> PASSED
 G19 : cut(average) =   891.44 /   906 ( 98.39% ), FPGA kernel time
(core time) = 0.006494930 ( 0.004349020 ) s -> PASSED
 G20 : cut(average) =   927.38 /   941 ( 98.55% ), FPGA kernel time
(core time) = 0.005637624 ( 0.003479216 ) s -> PASSED
 G21 : cut(average) =   918.63 /   931 ( 98.67% ), FPGA kernel time
(core time) = 0.010853278 ( 0.008698039 ) s -> PASSED
 G22 : cut(average) = 13268.96 / 13359 ( 99.33% ), FPGA kernel time
(core time) = 0.003015735 ( 0.000869804 ) s -> PASSED
 G23 : cut(average) = 13265.85 / 13344 ( 99.41% ), FPGA kernel time
(core time) = 0.003011570 ( 0.000869804 ) s -> PASSED
 G24 : cut(average) = 13257.14 / 13337 ( 99.40% ), FPGA kernel time
(core time) = 0.003012549 ( 0.000869804 ) s -> PASSED
 G25 : cut(average) = 13262.44 / 13340 ( 99.42% ), FPGA kernel time
(core time) = 0.003013803 ( 0.000869804 ) s -> PASSED
 G26 : cut(average) = 13256.80 / 13328 ( 99.47% ), FPGA kernel time
(core time) = 0.003013421 ( 0.000869804 ) s -> PASSED
 G27 : cut(average) =  3323.86 /  3341 ( 99.49% ), FPGA kernel time
(core time) = 0.010849928 ( 0.008698039 ) s -> PASSED
 G28 : cut(average) =  3277.72 /  3298 ( 99.39% ), FPGA kernel time
(core time) = 0.006497843 ( 0.004349020 ) s -> PASSED
 G29 : cut(average) =  3380.97 /  3405 ( 99.29% ), FPGA kernel time
(core time) = 0.010848867 ( 0.008698039 ) s -> PASSED
 G30 : cut(average) =  3387.46 /  3413 ( 99.25% ), FPGA kernel time
(core time) = 0.006499671 ( 0.004349020 ) s -> PASSED
 G31 : cut(average) =  3296.58 /  3310 ( 99.59% ), FPGA kernel time
(core_time) = 0.010846124 ( 0.008698039 ) s -> PASSED
```

```
 G32 : cut(average) =  1394.46 /  1410 ( 98.90% ), FPGA kernel time
(core time) = 0.019542269 ( 0.017396078 ) s -> PASSED
 G33 : cut(average) =  1367.28 /  1382 ( 98.93% ), FPGA kernel time
(core time) = 0.019546468 ( 0.017396078 ) s -> PASSED
 G34 : cut(average) =  1370.68 /  1384 ( 99.04% ), FPGA kernel time
(core time) = 0.028235555 ( 0.026094118 ) s -> PASSED
 G35 : cut(average) =  7628.58 /  7687 ( 99.24% ), FPGA kernel time
(core time) = 0.005638055 ( 0.003479216 ) s -> PASSED
 G36 : cut(average) =  7622.60 /  7680 ( 99.25% ), FPGA kernel time
(core time) = 0.005639398 ( 0.003479216 ) s -> PASSED
 G37 : cut(average) =  7632.13 /  7691 ( 99.23% ), FPGA kernel time
(core time) = 0.005638912 ( 0.003479216 ) s -> PASSED
 G38 : cut(average) =  7631.39 /  7688 ( 99.26% ), FPGA kernel time
(core time) = 0.005637739 ( 0.003479216 ) s -> PASSED
 G39 : cut(average) =  2378.19 /  2408 ( 98.76% ), FPGA kernel time
(core time) = 0.015198866 ( 0.013047059 ) s -> PASSED
 G40 : cut(average) =  2367.17 /  2400 ( 98.63% ), FPGA kernel time
(core time) = 0.015197196 ( 0.013047059 ) s -> PASSED
 G41 : cut(average) =  2367.43 /  2405 ( 98.44% ), FPGA kernel time
(core time) = 0.010851339 ( 0.008698039 ) s -> PASSED
 G42 : cut(average) =  2447.22 /  2481 ( 98.64% ), FPGA kernel time
(core time) = 0.015195246 ( 0.013047059 ) s -> PASSED
 G43 : cut(average) =  6619.69 /  6660 ( 99.39% ), FPGA kernel time
(core time) = 0.003010774 ( 0.000869804 ) s -> PASSED
 G44 : cut(average) =  6592.61 /  6650 ( 99.14% ), FPGA kernel time
(core time) = 0.002581100 ( 0.000434902 ) s -> PASSED
 G45 : cut(average) =  6613.87 /  6654 ( 99.40% ), FPGA kernel time
(core time) = 0.003010984 ( 0.000869804 ) s -> PASSED
 G46 : cut(average) =  6591.45 /  6649 ( 99.13% ), FPGA kernel time
(core time) = 0.002583113 ( 0.000434902 ) s -> PASSED
 G47 : cut(average) =  6601.29 /  6657 ( 99.16% ), FPGA kernel time
(core time) = 0.002581452 ( 0.000434902 ) s -> PASSED
 G51 : cut(average) =  3816.51 /  3848 ( 99.18% ), FPGA kernel time
(core time) = 0.004344999 ( 0.002174510 ) s -> PASSED
 G52 : cut(average) =  3820.76 /  3851 ( 99.21% ), FPGA kernel time
(core time) = 0.004345797 ( 0.002174510 ) s -> PASSED
 G53 : cut(average) =  3817.74 /  3850 ( 99.16% ), FPGA kernel time
(core time) = 0.003908539 ( 0.001739608 ) s -> PASSED
 G54 : cut(average) =  3821.00 /  3852 ( 99.20% ), FPGA kernel time
(core time) = 0.005205600 ( 0.003044314 ) s -> PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048INT2 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm v1 0 0/sbm/images/ISING C2048INT2 1 0.aocx
DEBUG : num devices 1
INFO : Time of fpga setup: 2.985635995 s
  G1 : cut(average) = 11539.31 / 11624 ( 99.27% ), FPGA kernel time
(core time) = 0.001135244 ( 0.000107962 ) s -> PASSED
  G2 : cut(average) = 11526.96 / 11620 ( 99.20% ), FPGA kernel time
(core_time) = 0.001129960 ( 0.000071975 ) s -> PASSED
```

```
  G3 : cut(average) = 11527.25 / 11622 ( 99.18% ), FPGA kernel time
(core time) = 0.001131011 ( 0.000071975 ) s -> PASSED
  G4 : cut(average) = 11526.66 / 11646 ( 98.98% ), FPGA kernel time
(core time) = 0.001081208 ( 0.000035987 ) s -> PASSED
  G5 : cut(average) = 11522.44 / 11631 ( 99.07% ), FPGA kernel time
(core time) = 0.001080978 ( 0.000035987 ) s -> PASSED
  G6 : cut(average) =  2168.53 /  2178 ( 99.57% ), FPGA kernel time
(core time) = 0.001403454 ( 0.000359873 ) s -> PASSED
  G7 : cut(average) =  1988.24 /  2006 ( 99.11% ), FPGA kernel time
(core time) = 0.001401666 ( 0.000359873 ) s -> PASSED
  G8 : cut(average) =  1986.44 /  2005 ( 99.07% ), FPGA kernel time
(core time) = 0.001242569 ( 0.000215924 ) s -> PASSED
  G9 : cut(average) =  2029.93 /  2054 ( 98.83% ), FPGA kernel time
(core time) = 0.001242370 ( 0.000215924 ) s -> PASSED
 G10 : cut(average) =  1978.85 /  2000 ( 98.94% ), FPGA kernel time
(core time) = 0.001243766 ( 0.000215924 ) s -> PASSED
 G11 : cut(average) =   559.14 /   564 ( 99.14% ), FPGA kernel time
(core time) = 0.001773897 ( 0.000719745 ) s -> PASSED
 G12 : cut(average) =   550.52 /   556 ( 99.01% ), FPGA kernel time
(core time) = 0.001773980 ( 0.000719745 ) s -> PASSED
 G13 : cut(average) =   574.70 /   582 ( 98.75% ), FPGA kernel time
(core time) = 0.001773231 ( 0.000719745 ) s -> PASSED
 G14 : cut(average) =  3039.72 /  3064 ( 99.21% ), FPGA kernel time
(core time) = 0.001237308 ( 0.000179936 ) s -> PASSED
 G15 : cut(average) =  3025.49 /  3050 ( 99.20% ), FPGA kernel time
(core time) = 0.001239076 ( 0.000179936 ) s -> PASSED
 G16 : cut(average) =  3028.73 /  3052 ( 99.24% ), FPGA kernel time
(core time) = 0.001294850 ( 0.000251911 ) s -> PASSED
 G17 : cut(average) =  3020.84 /  3047 ( 99.14% ), FPGA kernel time
(core time) = 0.001238470 ( 0.000179936 ) s -> PASSED
 G18 : cut(average) =   974.66 /   992 ( 98.25% ), FPGA kernel time
(core time) = 0.001346244 ( 0.000287898 ) s -> PASSED
 G19 : cut(average) =   891.61 /   906 ( 98.41% ), FPGA kernel time
(core time) = 0.001402753 ( 0.000359873 ) s -> PASSED
 G20 : cut(average) =   927.11 /   941 ( 98.52% ), FPGA kernel time
(core time) = 0.001347731 ( 0.000287898 ) s -> PASSED
 G21 : cut(average) =   918.33 /   931 ( 98.64% ), FPGA kernel time
(core time) = 0.001772418 ( 0.000719745 ) s -> PASSED
 G22 : cut(average) = 13267.86 / 13359 ( 99.32% ), FPGA kernel time
(core time) = 0.001131475 ( 0.000071975 ) s -> PASSED
 G23 : cut(average) = 13267.51 / 13344 ( 99.43% ), FPGA kernel time
(core time) = 0.001131168 ( 0.000071975 ) s -> PASSED
 G24 : cut(average) = 13260.14 / 13337 ( 99.42% ), FPGA kernel time
(core time) = 0.001132685 ( 0.000071975 ) s -> PASSED
 G25 : cut(average) = 13264.84 / 13340 ( 99.44% ), FPGA kernel time
(core time) = 0.001131922 ( 0.000071975 ) s -> PASSED
 G26 : cut(average) = 13256.76 / 13328 ( 99.47% ), FPGA kernel time
(core time) = 0.001132248 ( 0.000071975 ) s -> PASSED
 G27 : cut(average) =  3324.12 /  3341 ( 99.49% ), FPGA kernel time
(core time) = 0.001774008 ( 0.000719745 ) s -> PASSED
 G28 : cut(average) =  3276.24 /  3298 ( 99.34% ), FPGA kernel time
(core_time) = 0.001402539 ( 0.000359873 ) s -> PASSED
```

```
 G29 : cut(average) =  3381.02 /  3405 ( 99.30% ), FPGA kernel time
(core time) = 0.001774168 ( 0.000719745 ) s -> PASSED
 G30 : cut(average) =  3391.13 /  3413 ( 99.36% ), FPGA kernel time
(core time) = 0.001401921 ( 0.000359873 ) s -> PASSED
 G31 : cut(average) =  3293.64 /  3310 ( 99.51% ), FPGA kernel time
(core time) = 0.001772701 ( 0.000719745 ) s -> PASSED
 G32 : cut(average) =  1394.98 /  1410 ( 98.93% ), FPGA kernel time
(core time) = 0.002474579 ( 0.001439490 ) s -> PASSED
 G33 : cut(average) =  1367.20 /  1382 ( 98.93% ), FPGA kernel time
(core time) = 0.002473644 ( 0.001439490 ) s -> PASSED
 G34 : cut(average) =  1371.18 /  1384 ( 99.07% ), FPGA kernel time
(core time) = 0.003172511 ( 0.002159236 ) s -> PASSED
 G35 : cut(average) =  7631.08 /  7687 ( 99.27% ), FPGA kernel time
(core time) = 0.001346002 ( 0.000287898 ) s -> PASSED
 G36 : cut(average) =  7623.69 /  7680 ( 99.27% ), FPGA kernel time
(core time) = 0.001345553 ( 0.000287898 ) s -> PASSED
 G37 : cut(average) =  7632.64 /  7691 ( 99.24% ), FPGA kernel time
(core time) = 0.001344986 ( 0.000287898 ) s -> PASSED
 G38 : cut(average) =  7632.52 /  7688 ( 99.28% ), FPGA kernel time
(core time) = 0.001346971 ( 0.000287898 ) s -> PASSED
 G39 : cut(average) =  2376.18 /  2408 ( 98.68% ), FPGA kernel time
(core time) = 0.002100899 ( 0.001079618 ) s -> PASSED
 G40 : cut(average) =  2366.51 /  2400 ( 98.60% ), FPGA kernel time
(core time) = 0.002100202 ( 0.001079618 ) s -> PASSED
 G41 : cut(average) =  2366.48 /  2405 ( 98.40% ), FPGA kernel time
(core time) = 0.001773197 ( 0.000719745 ) s -> PASSED
 G42 : cut(average) =  2446.40 /  2481 ( 98.61% ), FPGA kernel time
(core time) = 0.002098914 ( 0.001079618 ) s -> PASSED
 G43 : cut(average) =  6614.74 /  6660 ( 99.32% ), FPGA kernel time
(core time) = 0.001130044 ( 0.000071975 ) s -> PASSED
 G44 : cut(average) =  6592.56 /  6650 ( 99.14% ), FPGA kernel time
(core time) = 0.001081654 ( 0.000035987 ) s -> PASSED
 G45 : cut(average) =  6616.14 /  6654 ( 99.43% ), FPGA kernel time
(core time) = 0.001132551 ( 0.000071975 ) s -> PASSED
 G46 : cut(average) =  6592.11 /  6649 ( 99.14% ), FPGA kernel time
(core time) = 0.001080667 ( 0.000035987 ) s -> PASSED
 G47 : cut(average) =  6598.98 /  6657 ( 99.13% ), FPGA kernel time
(core time) = 0.001080955 ( 0.000035987 ) s -> PASSED
 G51 : cut(average) =  3816.72 /  3848 ( 99.19% ), FPGA kernel time
(core time) = 0.001238336 ( 0.000179936 ) s -> PASSED
 G52 : cut(average) =  3820.72 /  3851 ( 99.21% ), FPGA kernel time
(core time) = 0.001239302 ( 0.000179936 ) s -> PASSED
 G53 : cut(average) =  3817.55 /  3850 ( 99.16% ), FPGA kernel time
(core time) = 0.001186997 ( 0.000143949 ) s -> PASSED
 G54 : cut(average) =  3821.04 /  3852 ( 99.20% ), FPGA kernel time
(core time) = 0.001293886 ( 0.000251911 ) s -> PASSED
DEBUG : Platform: Intel(R) FPGA SDK for OpenCL(TM)
DEBUG : Using 1 device(s)
DEBUG :      pac s10 dc : Intel PAC Platform (pac f100000)
DEBUG :
PATH=/home/test0212/work/sbm v1 0 0/sbm/images/ISING C8192INT2 1 0
DEBUG : Using AOCX:
/home/test0212/work/sbm_v1_0_0/sbm/images/ISING_C8192INT2_1_0.aocx
```

```
DEBUG : num devices 1
INFO : Time of fpga setup: 3.313207484 s
  G1 : cut(average) = 11539.31 / 11624 ( 99.27% ), FPGA kernel time
(core time) = 0.018521832 ( 0.001093750 ) s -> PASSED
  G2 : cut(average) = 11526.96 / 11620 ( 99.20% ), FPGA kernel time
(core time) = 0.018154396 ( 0.000729167 ) s -> PASSED
  G3 : cut(average) = 11527.25 / 11622 ( 99.18% ), FPGA kernel time
(core time) = 0.018157935 ( 0.000729167 ) s -> PASSED
  G4 : cut(average) = 11526.66 / 11646 ( 98.98% ), FPGA kernel time
(core time) = 0.017794427 ( 0.000364583 ) s -> PASSED
  G5 : cut(average) = 11522.44 / 11631 ( 99.07% ), FPGA kernel time
(core time) = 0.017792201 ( 0.000364583 ) s -> PASSED
  G6 : cut(average) =  2168.53 /  2178 ( 99.57% ), FPGA kernel time
(core time) = 0.021069763 ( 0.003645833 ) s -> PASSED
  G7 : cut(average) =  1988.24 /  2006 ( 99.11% ), FPGA kernel time
(core time) = 0.021067299 ( 0.003645833 ) s -> PASSED
  G8 : cut(average) =  1986.44 /  2005 ( 99.07% ), FPGA kernel time
(core time) = 0.019616388 ( 0.002187500 ) s -> PASSED
  G9 : cut(average) =  2029.93 /  2054 ( 98.83% ), FPGA kernel time
(core time) = 0.019617600 ( 0.002187500 ) s -> PASSED
 G10 : cut(average) =  1978.85 /  2000 ( 98.94% ), FPGA kernel time
(core time) = 0.019614449 ( 0.002187500 ) s -> PASSED
 G11 : cut(average) =   559.14 /   564 ( 99.14% ), FPGA kernel time
(core time) = 0.024713408 ( 0.007291667 ) s -> PASSED
 G12 : cut(average) =   550.52 /   556 ( 99.01% ), FPGA kernel time
(core time) = 0.024714101 ( 0.007291667 ) s -> PASSED
 G13 : cut(average) =   574.70 /   582 ( 98.75% ), FPGA kernel time
(core time) = 0.024713621 ( 0.007291667 ) s -> PASSED
 G14 : cut(average) =  3039.72 /  3064 ( 99.21% ), FPGA kernel time
(core time) = 0.019247668 ( 0.001822917 ) s -> PASSED
 G15 : cut(average) =  3025.49 /  3050 ( 99.20% ), FPGA kernel time
(core time) = 0.019249242 ( 0.001822917 ) s -> PASSED
 G16 : cut(average) =  3028.73 /  3052 ( 99.24% ), FPGA kernel time
(core time) = 0.019975660 ( 0.002552083 ) s -> PASSED
 G17 : cut(average) =  3020.84 /  3047 ( 99.14% ), FPGA kernel time
(core time) = 0.019248836 ( 0.001822917 ) s -> PASSED
 G18 : cut(average) =   974.66 /   992 ( 98.25% ), FPGA kernel time
(core time) = 0.020338645 ( 0.002916667 ) s -> PASSED
 G19 : cut(average) =   891.61 /   906 ( 98.41% ), FPGA kernel time
(core time) = 0.021072593 ( 0.003645833 ) s -> PASSED
 G20 : cut(average) =   927.11 /   941 ( 98.52% ), FPGA kernel time
(core time) = 0.020342400 ( 0.002916667 ) s -> PASSED
 G21 : cut(average) =   918.33 /   931 ( 98.64% ), FPGA kernel time
(core time) = 0.024716439 ( 0.007291667 ) s -> PASSED
 G22 : cut(average) = 13267.86 / 13359 ( 99.32% ), FPGA kernel time
(core time) = 0.018154012 ( 0.000729167 ) s -> PASSED
 G23 : cut(average) = 13267.51 / 13344 ( 99.43% ), FPGA kernel time
(core time) = 0.018152900 ( 0.000729167 ) s -> PASSED
 G24 : cut(average) = 13260.14 / 13337 ( 99.42% ), FPGA kernel time
(core time) = 0.018160259 ( 0.000729167 ) s -> PASSED
 G25 : cut(average) = 13264.84 / 13340 ( 99.44% ), FPGA kernel time
(core_time) = 0.018153328 ( 0.000729167 ) s -> PASSED
```

```
 G26 : cut(average) = 13256.76 / 13328 ( 99.47% ), FPGA kernel time
(core time) = 0.018151583 ( 0.000729167 ) s -> PASSED
 G27 : cut(average) =  3324.12 /  3341 ( 99.49% ), FPGA kernel time
(core time) = 0.024716923 ( 0.007291667 ) s -> PASSED
 G28 : cut(average) =  3276.24 /  3298 ( 99.34% ), FPGA kernel time
(core time) = 0.021068234 ( 0.003645833 ) s -> PASSED
 G29 : cut(average) =  3381.02 /  3405 ( 99.30% ), FPGA kernel time
(core time) = 0.024714991 ( 0.007291667 ) s -> PASSED
 G30 : cut(average) =  3391.13 /  3413 ( 99.36% ), FPGA kernel time
(core time) = 0.021072058 ( 0.003645833 ) s -> PASSED
 G31 : cut(average) =  3293.64 /  3310 ( 99.51% ), FPGA kernel time
(core time) = 0.024715193 ( 0.007291667 ) s -> PASSED
 G32 : cut(average) =  1394.98 /  1410 ( 98.93% ), FPGA kernel time
(core time) = 0.032002710 ( 0.014583333 ) s -> PASSED
 G33 : cut(average) =  1367.20 /  1382 ( 98.93% ), FPGA kernel time
(core time) = 0.032006323 ( 0.014583333 ) s -> PASSED
 G34 : cut(average) =  1371.18 /  1384 ( 99.07% ), FPGA kernel time
(core time) = 0.039294627 ( 0.021875000 ) s -> PASSED
 G35 : cut(average) =  7631.08 /  7687 ( 99.27% ), FPGA kernel time
(core time) = 0.020341601 ( 0.002916667 ) s -> PASSED
 G36 : cut(average) =  7623.69 /  7680 ( 99.27% ), FPGA kernel time
(core time) = 0.020338461 ( 0.002916667 ) s -> PASSED
 G37 : cut(average) =  7632.64 /  7691 ( 99.24% ), FPGA kernel time
(core time) = 0.020338859 ( 0.002916667 ) s -> PASSED
 G38 : cut(average) =  7632.52 /  7688 ( 99.28% ), FPGA kernel time
(core time) = 0.020341385 ( 0.002916667 ) s -> PASSED
 G39 : cut(average) =  2376.18 /  2408 ( 98.68% ), FPGA kernel time
(core time) = 0.028358871 ( 0.010937500 ) s -> PASSED
 G40 : cut(average) =  2366.51 /  2400 ( 98.60% ), FPGA kernel time
(core time) = 0.028357511 ( 0.010937500 ) s -> PASSED
 G41 : cut(average) =  2366.48 /  2405 ( 98.40% ), FPGA kernel time
(core time) = 0.024713721 ( 0.007291667 ) s -> PASSED
 G42 : cut(average) =  2446.40 /  2481 ( 98.61% ), FPGA kernel time
(core time) = 0.028354721 ( 0.010937500 ) s -> PASSED
 G43 : cut(average) =  6614.74 /  6660 ( 99.32% ), FPGA kernel time
(core time) = 0.018156961 ( 0.000729167 ) s -> PASSED
 G44 : cut(average) =  6592.56 /  6650 ( 99.14% ), FPGA kernel time
(core time) = 0.017791510 ( 0.000364583 ) s -> PASSED
 G45 : cut(average) =  6616.14 /  6654 ( 99.43% ), FPGA kernel time
(core time) = 0.018152410 ( 0.000729167 ) s -> PASSED
 G46 : cut(average) =  6592.11 /  6649 ( 99.14% ), FPGA kernel time
(core time) = 0.017793650 ( 0.000364583 ) s -> PASSED
 G47 : cut(average) =  6598.98 /  6657 ( 99.13% ), FPGA kernel time
(core time) = 0.017791359 ( 0.000364583 ) s -> PASSED
 G48 : cut(average) =  5953.78 /  6000 ( 99.23% ), FPGA kernel time
(core time) = 0.018153764 ( 0.000729167 ) s -> PASSED
 G49 : cut(average) =  5940.84 /  6000 ( 99.01% ), FPGA kernel time
(core time) = 0.018152865 ( 0.000729167 ) s -> PASSED
 G50 : cut(average) =  5834.06 /  5880 ( 99.22% ), FPGA kernel time
(core time) = 0.018155474 ( 0.000729167 ) s -> PASSED
 G51 : cut(average) =  3816.72 /  3848 ( 99.19% ), FPGA kernel time
(core_time) = 0.019250046 ( 0.001822917 ) s -> PASSED
```

```
 G52 : cut(average) =  3820.72 /  3851 ( 99.21% ), FPGA kernel time
(core time) = 0.019248623 ( 0.001822917 ) s -> PASSED
 G53 : cut(average) =  3817.55 /  3850 ( 99.16% ), FPGA kernel time
(core time) = 0.018880475 ( 0.001458333 ) s -> PASSED
 G54 : cut(average) =  3821.04 /  3852 ( 99.20% ), FPGA kernel time
(core time) = 0.019976614 ( 0.002552083 ) s -> PASSED
 G55 : cut(average) = 10222.49 / 10299 ( 99.26% ), FPGA kernel time
(core time) = 0.019616771 ( 0.002187500 ) s -> PASSED
 G56 : cut(average) =  3989.89 /  4017 ( 99.33% ), FPGA kernel time
(core time) = 0.046587996 ( 0.029166667 ) s -> PASSED
 G57 : cut(average) =  3462.20 /  3494 ( 99.09% ), FPGA kernel time
(core time) = 0.090331964 ( 0.072916667 ) s -> PASSED
 G58 : cut(average) = 19134.49 / 19293 ( 99.18% ), FPGA kernel time
(core time) = 0.019613959 ( 0.002187500 ) s -> PASSED
 G59 : cut(average) =  6019.58 /  6087 ( 98.89% ), FPGA kernel time
(core time) = 0.053877484 ( 0.036458333 ) s -> PASSED
 G60 : cut(average) = 14080.47 / 14190 ( 99.23% ), FPGA kernel time
(core time) = 0.019614184 ( 0.002187500 ) s -> PASSED
 G61 : cut(average) =  5745.69 /  5798 ( 99.10% ), FPGA kernel time
(core time) = 0.032006100 ( 0.014583333 ) s -> PASSED
 G62 : cut(average) =  4826.66 /  4870 ( 99.11% ), FPGA kernel time
(core time) = 0.090330541 ( 0.072916667 ) s -> PASSED
 G63 : cut(average) = 26827.61 / 27045 ( 99.20% ), FPGA kernel time
(core time) = 0.019615762 ( 0.002187500 ) s -> PASSED
 G64 : cut(average) =  8660.93 /  8751 ( 98.97% ), FPGA kernel time
(core time) = 0.090330809 ( 0.072916667 ) s -> PASSED
 G65 : cut(average) =  5509.26 /  5562 ( 99.05% ), FPGA kernel time
(core time) = 0.090332441 ( 0.072916667 ) s -> PASSED
Total FAILED count: 0
--- G,K,M check (end) ---
--- TSP check (start) ---
make: `all' に対して行うべき事はありません.
INFO : Time of fpga setup: 2.900103878 s
minimum distance: 704.00
The route is as follows
[ 16 15 14 13 12 11 10 9 8 7 6 5 4 2 3 0 1 41 40 39 38 37 36 35 34
33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 ]
dist( 704.00 ), FPGA kernel time (core time) = 0.006441823
( 0.004301158 ) s -> OK
--- TSP check (end) ---
```