Martin Hruby
Faculty of Information Technology
Brno University of Technology
Czech republic
e-mail: hrubym at fit.vutbr.cz

PSP-SOLVER USER GUIDE

Introduction

PSPSolver is a configurable optimizer of Resource-Constrained Project Scheduling Problems (RCPSP). The program is called PSP-Solver because it accepts tasks from a well-known library of RCPSP tasks, the PSPlib. However, the PSP-Solver might load any other format following the RCPSP specification.

The PSP-Solver users are expected to have a basic knowledge of RCPSP solving techniques and terminology.

Installation & run

PSP-Solver can be compiled on Linux/UNIX systems or with a appropriate support of GNU software (e.g. Cygwin). Compilation requires:

- GNU C++ 4.2 distribution or higher.
- Boost devel libraries (thread, base).
- Open MP.

Unpack the distribution (or SVN repository) and type "make". Compilation results with a binary file called "psp".

To run PSP-Solver, type

```
> ./psp -u <rcp_file> [-args]
```

where:

- rcp_file is a path to a RCPS problem in the RCP format (see PSPLib for more information).
- args is a sequence of simple boolean flags (-x) or input value settings (-x value).

RCPSP Model Specification

Task definition

A RCPS problem (task) is given by its:

- set of jobs J = {j0, j1, j2, ..., jN, jN+1},

- set of resources R = {r1,r2, ..., rM},
- capacity of resources C: R ➝ Integer,
- requested capacity for each job and resource K: J × R ➝ Integer,
- precedence relation P on J × J, where (i,j)∈ P means that j earliest start must be after i's end,
- duration of jobs D: J ➝ Integer; D(j0)=D(jN+1)=0.

Task solution

Solution of a given task is a vector S = (S(j0), ..., S(jN+1)), where S(j) is a starting time of a job j. A solution S is valid if it keeps task's definition, which is:

- precedence: forall (i,j) from P: S(j) >= S(i) + D(i),
- resource constraints: forall t from {0, 1, ..., S(jN+1)}, forall r from R: summary of K(i,k) of jobs running at the t-moment is not higher than C(k).

The main goal of all efforts in solving RCPS problems is to find a valid solution, and moreover, to find a good or event optimal solution in the meaning of some quality of the solution -- which is so called makespan here. Makespan of a particular solution S is the moment when the last jobs starts/ends, i.e. S(jN+1).

For the technical purposes, RCPS optimizer work with certain encodings of solutions. The most common encoding is called Activity List (AL). AL is an vector of jobs specifying an order of scheduling the jobs using a Schedule Generation Scheme (SGS).

As every AL put into a specified SGS determines a particular solution, ALs are usually presented as the solutions (keeping in mind that under some SGS).

PSP-Solver Program Architecture

The program loads a given task, creates a specified algorithmic solver, invokes the solver and prints its results (a solution). At the moment, PSP-Solver contains the following algorithms and solvers:

- Basic Genetic Algorithm with RT (GARTH) — configurable GA with or without RT support.
- Experimental Genetic Algorithm (E-GARTH) — highly configurable GA inspired from BGA.
- Multi-population Genetic Algorithm (MPGA) — alg. runs a user specifed number of GA optimizers and swap sub-sets of their genes at certain moments.
- Brute-force Solver — alg. enumerates all ALs.
- Histogram Scanner — randomly generates genomes, evaluates them an results a histogram of found makespans.

The GARTH, E-GARTH and MPGA solvers may be configured at each step of their functionality. GARTH and E-GARTH work generally in the following steps:

1. Creating an initial population — the population may be fully random or loaded from a given file.
2. Evaluation of the initial population and check halting conditions.
3. For a pre-defined number of generations (transform an old population into a new population):
    1. Select genomes for their copying into a new population.
    2. Copy selected genomes into the new population.
    3. Select genomes for mutation.
    4. Select genomes for crossover operation.
    5. Proceed the mutation operation and put the resulting genomes to the new population.
    6. Proceed the crossover operation and put the resulting genomes to the new population.
    7. Replace the old population with the new population.
    8. Evaluate the new population.
    9. Check halting conditions.
4. Export the best found solution. The result is printed on standard output and may be sent to a PostgreSQL database (if configured).

GARTH and E-GARTH slightly differs in order of the above specified steps. Multi-population GA (MPGA) instantiates a specified algorithm (GARTH or E-GARTH) such that all instances are fully independent (they do not share a pseudo-random generator).

RT Support

If RT (Run-time hypothesis) is enabled, the created optimizers manages a special set of statistics on their past evolution. A single RT statistics is a structure containing:

- a type of statistics — particular RT-property of two jobs (i, j) describing a sort of their mutual time constalation,
- a pair (i,j) of jobs,
- makespan — so far best makespan of a genome where (i,j) were at the specified mutual time constalation.

PSP-Solver contains various types of RT-properties, e.g. PSE (Paralel Starts Equal) meaning that, in a certain AL (genome), jobs i and j are in PSE time mutual constalation if they start at equal time moment. A RT-statistics (PSE, (i,j), M) then says that during the past GA evolution, there was not a genome with (i,j) in PSE mutual constalation having its makespan better than M. Having such a RT-statistics, the GA may assume a hypothesis that (i,j) in PSE blocks a solution better than M, or better say there is probably no AL with (i,j) starting in the same moment which would have a makespan better than M.

A single RT-statistics is just a hypothesis, not a fact. It helps the BGA (EGA) to spread its evolution over the whole possible space of all solutions.

RT-characteristics are being collected on population re-evaluation. Every evaluated genome extracts its RT-characteristics and these are put to a global RT data structure (RT-System).

RT-System gives a useful insight to evolution of a GA and also helps to select jobs for their shifts within an AL (mutation).

PSP-Solver User Options Overview

PSP-Solver is a command-prompt simple program with no GUI. All options are entered via program command line arguments:

```
./psp [options]
```

Option format	Name	Default value
-2	When crossing two genomes, use both resulting children.	False.
-1	Allow parallel run of program.	Disabled.
-@ filename	Set of ALs for population init.	Not used.
-# int	Maximum number of genome evaluations.	Not limited.
-$ string	Auxiliaty string.	Empty
-a	Create new ALs by RT-specific mutations	False.
-A float	Ratio of newly rnd generated genomes. Values in <0,1>.	0
-c int	Maximum number of mutated jobs in one AL.	10
-C int	Cross partners selection method.	0

Option format	Name	Default value
-d	Proceed a small mutation on a genome created by crossover.	False.
-D int	Muta selection method.	0
-E ID	Experiment identification string.	Empty
-G int	Basic algorithm number (use just with MPGA).	0
-g int	Number of generations.	
-I int	Auxiliaty integer.	
-K int	Mutation description mode.	0
-L	Load/Save RT	False.
-l int	Maximum numbers of LR-justification cycles.	
-m int	RT selection mask.	5
-M int	Program run mode.	0
-N	Reverse optimization sense (find maximum on the objective function).	False.
-n int	Pseudo-random generator initialization method (seed).	3
-p int	Number of genomes in population.	
-P int-int-int	PSP task ID (group, set, item)	30-1-1
-Q int	Set mode of dynamic mutation cycles.	Fixed number of cycles.
-S int	AL SGS mode.	0
-s int	Maximum mutation depth.	Unlimited.
-T int	Selection for copy method.	
-U	Ignore resource capacities.	False.
-u filename	RCP filename to load.	
-V	Variability (MPGA)	False.

Option format	Name	Default value
-w	Disable RT.	RT enabled.
-W "list of float"	Configure crossover in E-GARTH.	TODO
-x	Seljob inverted	
-X int	Task number.	
-Y float	Ratio of mutated genomes in population.	0.2
-z	Disable AL normalization.	Enabled
-Z "list of int"	Fixed set of selected jobs for mutation.	Not used.

Detail description of program execution and behavior options

Loading the task

PSP-Solver loads tasks in RCP format which is used in PSPlib (data sets j30, j60, j120). The input task may be given by one of the following ways:

- -u filename,
- -P group-set-id, where group is in {30, 60, 120},
- -X num (num is in {1,…,480 or 600}).

In experimental mode, -E id sets an experiment identification string. The program results are then printed with a set experiment id. See section (...) for details on Experimenting.

Basic GA parameters

Genetic algorithms (GA) work over a set of N genomes (parameter **-p N**) and make M generation cycles (parameter **-g M**). If the task is entered with **-P** or **-X** option, PSP-Solver loads an information about best known solution (Upper Bound - UB) of a given task.

GA will stop at a generation when at least one genome reaches a makespen equal to the loaded UB. Add **-f** option if the GA should continue in further evolution.

At each generation cycle, GA selects some genomes for their copying without any modification (**-F float**), certain portion of the population for mutation (**-Y float**, where float in <0, 1>) and certain portion of the population to be replaced by newly randomly generated genomes (**-A float**). The rest of the population is then filled with genomes made by crossover operation.

Each constructed genome (randomly generated, mutated, made by crossover) is evaluated with a specified schedule scheme. The scheme is set with a **-S num** option, where num means:

- num = 0, standard serial generation schedule scheme (S-SGS).
- num = 5, S-SGS with repetitive Left-Right justification - MLR-SGS scheme.

Schedule generation scheme other than S-SGS interpret the genome's AL in generaly different order. For this reason, PSP-Solver normalizes genome's AL (see Genome normalization) if **-z** option is not set (i.e. normalization is default).

Program run basic parameters

PSP-Solver instantiate a solving algorithm or technique specified by **-M num** option. There are the following algorithms implemented:

Num	Name	Other specific options
0	Simple GA with random mutations	
1	GARTH	RT options
3	Brute force search	
4	Super BF	
5	Multi-Population GA	-G alg, -I popNum, -V
10	E-GARTH	
101	Histogram scanner	

The selected algorithm is generally an optimizer minimizing the objective criteria, which is normally a genome's makespan. For a study purposes, PSP-Solver allow searching for a maximum makespan if **-N** option is set.

Every algorithm instantiates its internal pseudo-random generator (Mersenne Twister) which is initiated with a seed specified by **-n num** option:

- num = 0, no initialization.
- num = 1, seed is taken from the OS clock information (not suitable for parallel instantiation of more rnd generators).
- num = 2, seed is taken from /dev/random.
- num = 3, seed is taken from /dev/urandom (default).
- num = 4, seed is taken from random.org (not finished yet).

GA terminates when/if:

- best known solution was found (if such a solution is available) and **-f** is NOT set, or
- predefined number of generations (**-g num**) was done, or
- predefined number of SGS was computed (**-# num**).

Evaluation of the population (computing SGS for each new genome) is the most time-consuming operation, thus PSP-Solver offers to make it in parallel. Add **-1** option to run this concurrently (you should run it on multi-core CPU then). Paralelism is implemented via OpenMP library.

The initial population is generated randomly by default. For an experimental purposes, it is also possible to insert a file with a set of ALs (**-@ filename** option). The population is then constructed from this file (up to a predefined **-p num**) and further filled up with random ALs up to the population size.

Select-for-copying specific options

The current population is ordered following the program's specific optimized criteria (minimum makespan by default). At each generation, a certain percentage of genomes (i.e. N genomes) is copied with no modification to the just being created new population. Selection of these genomes is driven by **-T num** option:

- num = 0, the best N genomes are selected.
- num = 1, N genomes are randomly selected.
- num = 2, N genomes with best combination of their objective and age are selected.
- num = 3, program evaluates a statistics on genomes and selects N mutually most distinct genomes.

Crossover operation specific options

These options solve a crossover specific problems:

- selection of two genomes to be combined (parents) with a two-point AL cross operator (option **-C num**),
- generation of the points c1 and c2 for the two-point cross operator (option **-O num**),
- inserting the resulting children to the new population (options **-d** and **-2**).

Let us assume two parents P1 and P2. Algorithm generates cross points c1 and c2 and proceeds the cross operator over P1 and P2 resulting with children CH1 and CH2. By default, only CH1 is inserted to the new population. If **-2** is set, then both CH1 and CH2 are inserted. Before their insertion, a so called small mutation may be applied on children if **-d** option is set.

The small mutation on a given AL means:

- selecting randomly a predefined number of jobs (=3 by default) -> selected jobs.
- for each selected job:
  - shift the job on left or on right (decided randomly for each one) within the AL, but for maximally N possitions in AL (=10 by default).

The parents P1 and P2 are selected randomly from subsets SUB1 and SUB2 of the current population depending on a particual GA algorithm.

In the Basic GA (-M 0, -M 1), the parent P1 is randomly chosen from the the whole old population and P2 is chosen to be his partner following the **-C num** setting:

- num = 0, P2 is chosen randomly.

- num = 1, P2 is chosen as most distant AL to the P1.
- num = 2, similar to num=1.

In the EGA algoritm, the amount of genomes decided to be created by crossover is split into 4 parts:

1. P1 is from genomes selected-for-copy, P2 is from newly created genomes (new randomly generated, mutated).
2. P1 is from newly created genomes, P2 is from the whole old population.
3. P1 is from the just being created new population, P2 is from the old population.
4. P1 and P2 are both randomly chosen from the old population.

Splitting the capacity for crossed genomes is user-specified by **-W float-list** option.

Decision on c1 and c2 crossing points is set by **-O num** option as follows:

- num = 0, c1 and c2 are randomly decided.
- num = 1, half-radnom (TODO).
- num = 2, set differs more than.
- num = 3, vector differs more than.

Mutation specific option

Mutation is the most complex operation in PSP-Solver. The mutation has the highest impact on diversity of populations being evolved.

By saying mutation of a job **j** on a possition **p** within a given **AL**, we usually mean shifting j from its starting possition p on left or on right with keeping j-job's precedence criterias, i.e. there is no job **k** to the left from j's current possition which is j's successor (there is no precedence (j,k)).

There are the following options regarding a single one AL to be mutated:

Option	**Name**	**Meaning**
-c num	Number of mutation cycles	"num" jobs in AL will be selected for shifting.
-s num	Maximum mutation depth	No job will be shifted from its starting possition **p** to a possition **r**, where |r-p|>num.
-D num	Method selecting genomes to be mutated.	
-Q num	Dynamic number of mutation cycles.	Code for a method dynamically deciding number of mutation cycles.
-K num	Mutation method	

At each generation, GA decides a number **N** of genomes to be mutated (**-Y float**). The algorithm then selects N genomes from the old population using a selection mechanism (**-D num**), where "num" means:

- n = 0, randomly chosen.
- n = 1, most wide spectrum of RT (to be explained...).
- n = 2,3,4,5,...

Selection of genomes results with a certain subset of genomes. For each one, the algorithm decides particular jobs to be shifted regarding the **-K num** option:

- num = 0, basic RT selection.
- num = 6, randomly chosen.
- ....

Solvers and tools

Histogram scanner (-M 101)

Relevant options:

- -G 8, to set Mendes-style genome (AL is default).
- -I num, num is a number of generated genomes.
- -S num, AL SGS code (-I num in addition to -S 5).
- -1 to run it in parallel.