RUNNER v.1

User manual

Program description

RUNNER is a program that performs a global search for the optimal structure of atomic clusters and the structure of their low-lying isomers, during which the cluster energy is evaluated using the quantum chemical (DFT or *ab initio*) or classical interatomic potential / force field calculations.

RUNNER carries out the global minimum search using the specialized server programs for energy evaluation and local optimization of the cluster geometry. Currently, the following programs can be servers for calculating energy and local optimization:

- Gaussian 03/09/16
- NWChem v. 6/7
- GLOBUS v.1

The global search method implemented in the program is the evolutionary (genetic) algorithm combined with a taboo search. The detailed description of global optimization algorithm implemented in the program is given in the section *Global optimization algorithm implemented in RUNNER*.

Obtaining and installing the program

The program, control script and examples of input files can be downloaded from the resource https://github.com/skignatov/runner-release

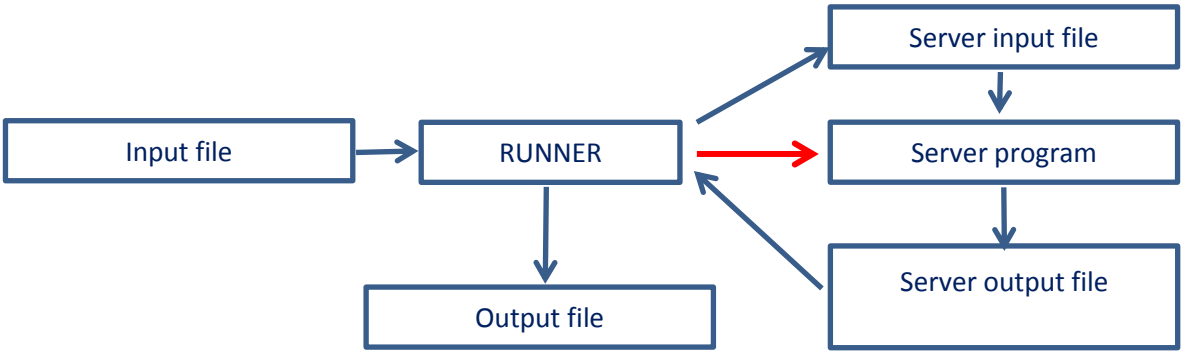
To get started, it is enough to create a separate directory on your computer, place the program, the control script (if necessary, see the section *Program operation modes*) and data files required for the selected operation mode (see the section *Files required for running*). In addition, the selected server program must also be correctly installed on the computer being used (or computers playing the role of remote computing nodes) (obtained and installed separately). The RUNNER start commands are described in the section *Running a program*.

Operating modes of the program

The program can work in several modes:

1. *Direct call mode* (Windows). In this mode, the program reads the main input data file, sequentially generates initial data files for the server program, and starts the server program on the computer being used to calculate with the generated file. After the end of

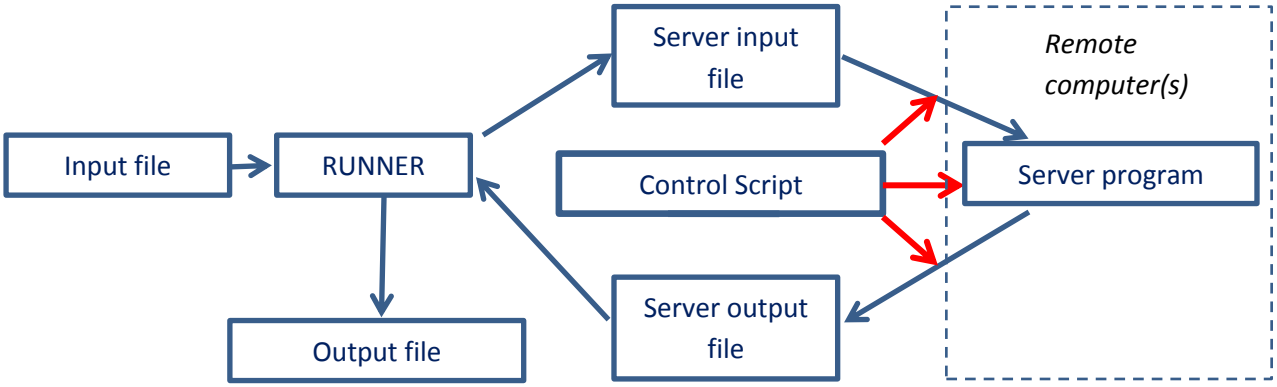
the calculation, the output file of the server program is analyzed, and information about the structure and energy of the cluster is extracted from it:



The direct call mode implements only sequential calculation of the structures under study. It is recommended to be used only for those calculations in which the calculation of energy takes a short time (for example, calculation based on empirical potentials and force fields by the GLOBUS program, quantum chemical calculation by the semiempirical method). This mode has been tested only under Windows.

2. *SSH mode* (LINUX). In this mode, the program reads the input data file, generates input data files for the server program (several files in parallel, the number of which is controlled by the MaxJobs command). The generated files are placed in the ./inp directory, which is automatically created in the current directory. A concurrently running control script (CS) *runner_ssh_ <version> .sh* reads files in the *./inp* directory and starts server calculations on one or several remote computers (nodes) that are connected to the central node via the ssh protocol. The central computer can also act as a remote site, i.e. tasks can be run on it as well. Upon successful launch of this file, the CS moves it from *./inp* to the *./out* directory, while creating a copy in the *./sub* directory. Files in this directory *./sub* are named *<file>. <ext> _X*, where *X* is the number of the node on which the calculation is started. The control system cyclically checks the status of calculations at remote nodes and upon completion of the calculation (successful or emergency) downloads the results file to the *./out* directory, deleting the corresponding file from *./sub*. RUNNER, detecting a new output file in the *./out* directory, parses it, extracting optimized structures and energies, and moves the input and output files from *./out* to the *./arc* directory. Depending on the reason for completing this calculation, RUNNER generates a new input file to continue the unfinished calculation or proceeds to generate a new point for global search. The generated files have the format of the name *geo-*XXXXXX-YY. *<ext>*, where XXXXXX is the number of the global search point, YY is the number of the continuation of the unfinished calculation (YY = 00 for a new file). The list of nodes on which server programs can be launched and calculations can be performed is contained in the nodes.txt file, which must be present in the startup directory. The CS can be started and stopped manually regardless of RUNNER (before or after it), or it can be started, restarted (in case of an emergency shutdown of the CS) and stopped by the program automatically (see the ShellControl command). SSH mode is recommended for tasks with long-term energy calculation (DFT, *ab initio*) on

workstations that are not combined into a cluster. Since quantum chemical calculations are performed in parallel at remote nodes, an increase in the number of available nodes increases the overall performance of GO.



3. *BATCH mode.* (LINUX on supercomputers or clusters). In this mode, the operation is completely similar to the SSH mode, but the control script *runner_batch_<version> .sh* starts tasks on a supercomputer (SC) or cluster using the task scheduler - *slurm* or *pbs*. The control script for *pbs* is different from the *slurm* script and is called *runner_batch_pbs_<version> .sh*. Unlike SSH mode, BATCH mode does not require a *nodes.txt* file, but two other files are required: (1) file *workdirs.txt*, which contains a list and description of the working directories where the results of individual tasks will appear, and (2) files of type *batch.template* which are forms for generating scripts for batch launching of computational tasks, formed according to the rules of the SC used. Files *batch.template* can be different and/or have different names for each working directory. Their respective names are specified in *workdirs.txt* for each working directory. When working in this mode, one or several directories are created in the program launch directory (their names are described in *workdirs.txt*, usually *wd0*, *wd1*, *wd2* ...) so that tasks for calculating individual global search points are launched in each directory in parallel. These directories also contain the output files of this calculation and all temporary files of the server program. The CS analyzes the files in the working directories and, upon completion of the calculation, moves them to the *./out* directory. After that, a new task is launched in this working directory, the input file of which is taken from the *./inp* directory. This mode of operation is the main one and the most convenient when calculating on the SC. An increase in the number of working directories leads to an acceleration of the calculation only within certain limits, since a large number of simultaneously running tasks can lead to the fact that some of them will be queued or will be blocked by the scheduler due to exceeding the task limit for one user. Often, 8-12 working directories are optimal for a single system.

4. *CRON mode, the batch mode with periodic start of control script.* (LINUX on supercomputers). On some SCs, the administrator prohibits continuous operation of the control script on a central node. To work on such computers, the CRON mode is used, which is completely similar to the BATCH mode, however, the control unit and the RUNNER program do not work continuously, but are started periodically by the *cron* command with a period of several minutes. The *cron* setting is performed by the control

script itself at the initial startup. This mode is less stable than the BATCH mode and it is recommended to use it only in case of strict restrictions on the operation of the control script.

Files required for running

Hereinafter, global optimization of the Mg_{10} cluster is considered as an example. All examples use *mg10* as the file name. During operation, this name should be replaced with the file name corresponding to a specific task.

File names and other parameters specified in <...> can vary at the request of the user. Files without such brackets must have names that exactly match those indicated in the table.

All data files must be in the startup directory. Programs can be located in directories to which the search path is set.

1. *Direct call mode*

File	**Description**
runner.exe (Windows) *runner.x* (LINUX)	Program RUNNER
<mg10>.inp	The main input data file. Describes the system to be optimized and optimization parameters
<mg.gj0>	Server program data file template

2. *SSH mode*

File	**Description**
runner.x	Program RUNNER
runner_ssh_<version>.sh	Control script
<mg10>.inp	The main input data file. Describes the system to be optimized and optimization parameters
<mg.gj0>	Server program data file template
nodes.txt	A file describing the ssh nodes on which the server programs will run.

3. *Режим BATCH*

File	**Description**
runner.x	Program RUNNER
runner_batch_<version>.sh	Control script (scheduler *slurm*). If *pbs* is used as a scheduler the script name is *runner_batch_pbs<_version>.sh*
<mg10>.inp	The main input data file. Describes the system to be optimized and optimization parameters
<mg.gj0>	Server program data file template
workdirs.txt	A file describing the directories in which individual tasks will be run. For each directory, its name is indicated, whether it is active (whether it is participating in the work at this time) and the name of the form of the

	task launch script.
<batch.template>	Form (template) of the task launch script. There can be several such files, they can have different names. These files are assigned to separate working directories in the *workdirs.txt* file

4. *Режим CRON*
   All files are completely identical to BATCH mode, except for the control script, which in this case is called *runner_batch_cron <_version> .sh* (for *slurm* scheduler).

Data files required for RUNNER operation

1. *The main file of the input data of RUNNER <mg10>.inp*

The main input data file for RUNNER is an ASCII text file. Keywords are located on lines, the first nonblank character of which is #. Blank lines and lines without a leading # are ignored. Symbol ! means the beginning of a comment, it can be located anywhere on the line. All information on this line to the right of ! is considered a comment.

The order of keywords, both on a line and on different lines, can be any. Uppercase and lowercase letters are the same (except for chemical symbols in the *Stoichiometry* command)

Basic commands and keywords (values in brackets <…> are examples):

Keyword	**Meaning**	**Default value**
Stoichiometry=<*Mg10*>	Sets the stoichiometry of the cluster to be optimized. The symbols of a chemical element should be written as given in the periodic table. Uppercase and lowercase letters are different, so it is wrong to write mg10 or MG10, only Mg10 is correct.	There is no default, the command must always be present
Nstruct=<100>	The number of global search cycles, i.e. the number of generated and optimized structures. Only those structures are considered optimized if the optimization has successfully reached the stopping criteria. The total number of optimizations, some of which did not reach the end or ended abnormally, may exceed *Nstruct*.	100
MaxCyc=<10000>	The number of work cycles performed by the RUNNER program while browsing the working directories. Prevents infinite looping if the number of optimized structures cannot reach *Nstruct* for some reason. The maximum running time of the program is *MaxCyc*IdleTime*	100000
IdleTime=<120> Idle=<120>	Waiting time at the end of each working cycle, in seconds	300
Server=<g09>	A type of server program that performs local optimization.	g09
FilPat=<*mg.gj0*>	The name of the template file of the server	No default value

	input file. The *%GEO%* label should be placed at the geometry placement. If the server is Gaussian, the *%chk=...* parameter must be present.	
Seed=(<12345>,<67890>)	Seed for initializing the random number generator. If there is no *Seed* command, the generator's internal defaults will be used. Used to repeat the results of a previous calculation.	Internal defaults of the random number generator
MaxPool=<30>	The maximum pool size. When it is exceeded, the found local minima, the energy of which exceeds the values of all structures in the pool, cannot participate in the generation of new structures.	50
ActivePool=<0.5>	The active part of the pool, participating in the generation of new structures.	0.7
MinPool=<5>	The minimum pool size at which the main GD phase begins	10
MaxRestarts=<20>	The maximum number of restarts of the local optimization (LO) of the server program, i.e. launches of the LO that were interrupted due to an excess of the number of cycles, internal errors, etc. Introduced to prevent endless server work when repeating the same error. If the number of restarts is exceeded, the structure is placed in the *./hlp* directory. This structure is ignored by *Nstruct*.	10
NoSaveArchive	Disable saving the results of the server program in the archive. In normal operation mode, each time the LO is completed by the server program, the input and output files of this optimization (for example, **.gjf* and **.log*) are sent to the directory *./arc* (archive of optimization results). If the files are very large and disk space is limited, the *NoSaveArchive* command cancels the save, saving disk space.	Missing (results are saved in the ./arc directory)
GenOnly	In *Serial mode*, do not run a quantum chemical calculation, only generate input files	Missing (start QC calculations)
BondLimits=(<1.>,<4>)	Limits of interatomic distances in Å at which atoms are considered to be bonded. Used when generating new structures by generation operators.	1 Å – 4 Å
Sphere=7.	The radius of the region in Å in which the atoms are generated by the operator RAND	7 Å
MaxTry=10000	The maximum number of attempts to generate a cluster in which the link lengths satisfy *BondLimits*	5000
Init=(<RAND>,<RND1>) Init=(<RAND=30,RND1=70)	Operators used to generate the initial population in the Initial Phase. The numbers after = indicate the likelihood of using the given operator. If the probabilities are not specified, they are considered equal (i.e. for	RAND, RND1

	two operators - 50% and 50%). For the list of available operators, see the section *Optimization Algorithm*.	
Operators=(<PCCR=50, RAND=10,RND1=10,TPCR=20,IMOV=10>)	Operators used to generate the initial population in the Main Production Phase. The numbers after = show the probability of using this operator (in%, automatically normalized to 100). If the probabilities are not specified, they are considered equal for all operators for which the probabilities are not specified. For a list of available operators, see the *Optimization Algorithm* section.	PCCR=50, RAND=10, RND1=10, TPCR=20, IMOV=10
Continue	Continue the previously completed optimization (i.e., an optimization that has reached the Nstruct limit). To continue, the files **.base_*, **.pool_*, **.sbas_*, **.info_* of the previous optimization will be used, they must be located in the startup directory. With the *Continue* command, the *Nstruct* parameter is treated as an additional number of structures to be examined during the ongoing optimization.	Missing (start new optimization)
ShellControl= (*StartOnBeg*,*StartIfDown*, *StopOnFinal* ,*StopOnExit*)	Management of starting, restarting and stopping the control script (CS). *StartOnBeg* – start CS in the beginning. *StartIfDown* – restart CS if it is stopped/failed. *StopOnFinal* – stop CS after optimization. *StopOnExit* – stop CS on the program exit.	If any keyword (or the entire command) is missing, its action is not taken.

In this and subsequent sections, the tags and keywords shown in red in the examples must be present in the files. Other keywords and commands are optional.

Sample file <Mg10>.inp:

```
# Stoichiometry=Mg10   Nstruct=300
# Server=g09   FilPat=mg.gj0   Idle=60  NoAutoStart
!# SEED=(674127515,48622252)

# MaxPool=50 ActivePool=0.7 MinPool=10 MaxRestarts=10  !NoSaveArchive
# BondLIMIts=(1.,4.) Sphere=6.  MaxTry=5000

# Init=(RND1=10,RAND=10)
# Operators=(PCCR=50,RAND=10,RND1=10,TPCR=20,IMOV=10)
```

2. *Template for server input file <mg.gj0>*

The file template for creating the input file of the server program is a slightly modified regular file of this program, in which the place where the Cartesian coordinates are located is marked with the *%GEO%* label. In addition, the *%chk*=TTT keyword (TTT is any text) must be present

in the files for the Gaussian program. During operation, the TTT text will be automatically replaced with the name of the temporary file *geo*-XXXXX-YY.*chk*, where XXXXXX and YY correspond to the structure number and restart number during its optimization.

Sample file:

```
%Nproc=4
%Mem=6000MB
%chk=geo.chk      ← required keyword %chk
%NoSave
# bp86/6-31G(d,p) Fopt=(MaxCycle=300)  SCF=(xqc,MaxCycle=300)

gjf-pattern file

0 1
%GEO%             ← required label %GEO% : coordinates are placed here
```

3. *File nodes.txt*

The file is an ASCII text file in which each line describes the node where the server program is running. Blank lines and lines starting with the # character are ignored. The part of the line after the # is treated as a comment.

Each line has the format:

<IP > <Active> <User> <Nproc> <Mem> <Prog> <Dir>

IP	The IP address of the node at which the node is accessible via *ssh*
Active	Whether the node will be used in the current optimization (0 or 1)
User	*Username* for *ssh* connection
Nproc	The number of cores that can be used in the calculation on a given node
Mem	The amount of memory in MB that can be used on this node
Prog	Server program start command
Dir	Directory in which the calculation will be carried out

Sample file:

```
# ip          Active  User      Nproc   Mem     Prog    Dir
45.3.23.189    1     user1      4    16000MB   g16    /home/user1/Documents/gwork     # 0
45.3.23.190    1     user1      4    16000MB   g16    /home/user1/Documents/gwork     # 1
45.3.23.135    1     student    4    16000MB   g16    /srv1/gwork                     # 2
45.3.23.158    1     student    4    16000MB   g16    /home/student/Documents/gwork   # 3
45.3.23.157    1     student    4    12000MB   g09    /home/student/Documents/gwork   # 4
45.3.23.153    1     student    8     6000MB   g09    /home/student/Documents/gwork   # 5
45.3.23.167    1     student    4     6000MB   g16    /srv1/gwork                     # 6
45.3.23.165    1     student    4     6000MB   g09    /home/student/Documents/gwork   # 7
45.3.23.131    1     student    4     6000MB   g09    /home/student/Documents/gwork   # 8
45.3.23.134    1     student    4     6000MB   g09    /home/student/Documents/gwork   # 9
45.3.23.143    0     student    4     6000MB   g09    /home/student/Documents/gwork   # 10
45.3.23.144    1     student    4     6000MB   g09    /home/student/Documents/gwork   # 11
```

```
45.3.23.148   1    student   4    6000MB   g09    /home/student/Documents/gwork   # 12
```

4. *Файл workdirs.txt*

The file is an ASCII text file, in which each line describes the directory in which the job will be launched using the BATCH method using *slurm* or *pbs*. Blank lines and lines starting with the # character are ignored. The part of the line after the # is treated as a comment.

Each line has the format:

<Dirname > <Active> <No.> <Script>

Dirname	Directory name
Active	Whether the node will be used in the current optimization (0 or 1)
No.	The number of the directory. Does not affect the work, used as a comment.
Script	BATCH script template used to run jobs in this directory. Based on this template, the CS creates the final batch-script in the given directory. The template must contain the tags %%% and @@@ which are replaced by CS with, respectively, the unique name of the task and the input file of the server program.

Sample file:

```
#Dirname  Active?    No.   Script
wd0         1         0      batch.run0
wd1         1         1      batch.run0
wd2         1         1      batch.run0
wd3         1         3      batch.run0
wd4         1         4      batch.run0
wd5         1         5      batch.run0
wd6         1         6      batch.run0
wd7         1         7      batch.run0
wd8         1         8      batch.run0
wd9         1         9      batch.run0
wd10        1        10      batch.run_skx
```

5. *File <batch.template>*

The template file for creating a script for launching tasks in the BATCH and CRON modes is a slightly modified regular script of this program, in which the place where the name of the task is located is marked with a %%% mark, and the place where the name of the input file of the server program is placed is marked with @ @@. In the course of work, the text %%% will be automatically replaced with the task name in the form SXXXXX, where S is the letter specified by the user in the *jletter* parameter in the control script file (see the section *Starting the program*), and XXXXX corresponds to the structure number during its optimization. Using different *jletter*=”<S>” parameters for different systems to be optimized allows you to distinguish between the concurrent jobs displayed by the *squeue* and *qstat* commands.

*Sample file (*for *slurm* scheduler*):*

```
#!/bin/bash
#SBATCH -J %%%          ← required label %%% of the job name
#SBATCH -o %%%.o%j      ← (there can be multiple ones)
#SBATCH -e %%%.e%       ← (there can be multiple ones)
#SBATCH -p normal
#SBATCH -N 1
#SBATCH -n 63
#SBATCH -t 12:00:00

module load gaussian/16rA.03

InputFile=@@@           ← required label @@@ of the input file name

g16 ${InputFile}
```

*Sample file (*for *pbs* scheduler*):*

```
#!/bin/bash
#PBS -N %%%             ← required label %%% of the job name
#PBS -l nodes=1:ppn=12

export GAUSS_LFLAGS='-opt "Tsnet.Node.Lindarsharg:ssh"'
cd $PBS_O_WORKDIR

inputfile=@@@           ← required label @@@ of the input file name
output=${inputfile%.gjf}.log

g09 <$PWD/${inputfile} >$PWD/${output}
```

Starting the program

1. *Direct call mode*

Run command (Windows):

```
runner.exe mg10.inp
```

2. *SSH mode*

The command to run the program in the background:

```
./runner.x mg10.inp &
disown
```

Control script (CS) is launched automatically or manually before or after the program starts, depending on the *ShellControl* parameter (see the section *Input file keywords*). When starting manually, the command to start CS is:

```
./runner_ssh.sh >res.log
```

At startup, the RS asks for passwords for all remote nodes, which must be entered by the user in response to requests:

```
>Password for node 1, user student:  absde123
>Password for node 2, user user1:    jhkk12h
…
```

After entering the passwords, the control system can be switched to the background mode:

```
Ctrl-Z
bg
disown
```

During the operation of the CS, the *res.log* file will contain updated information about its operation.

WARNING! Some versions of CS do not ask for passwords! They should be indicated inside the script body.

3. *BATCH mode*

In this mode, it is recommended to start control script first, which then automatically starts the program. Sample run command is (here, *_e30* is the script version which can vary):

```
./runner_batch_e30.sh >res.log &
```

Before starting the CS, it should be edited using any editor (*mc, vi* ...), setting several mandatory options described at the beginning of the *runner_batch.sh* file:

```
########################################
#### Define your defaults here: #########
########################################

# These are important parameters! Check it carefully!
user="student"          ← indicate your login name (username)
InputFile=mg7.inp       ← indicate the runner input file
jletter="y"             ← indicate the latter to distinguish the jobs
curdir=$PWD             ← optional: work directory if different from current
idle_time=300           ← optional: sleep time, s
maxcyc=1000000          ← optional: maximum number of CS wakeups (cycles)

#for PBS only  (prefix added by qsub to JobNo)
compname=".master1.cyberia.tsu.ru"  ← for pbs: suffix of job number
                                      as indicated in the qstat command
```

4. *CRON mode*

Launching the program (as well as configuring the CS before launching) is completely similar to the BATCH mode. Run command is

```
./runner_batch_cron.sh >res.log &
```

Global optimization algorithm implemented in RUNNER

The global optimization algorithm implemented in RUNNER is based on an evolutionary (genetic) algorithm combined with a taboo search. The genetic algorithm is based on the application of generating operators (generators) to the initial and updatable pool of structures to create descendant structures. The “best offspring” selection method is to displace the pool structures by optimized structures with higher fitness (lower total energy).

There are 10 types of generators implemented in the program:

RAND	Random generation of atoms in a region of a size set by the *Sphere* command
RND1	Generation of atoms by condensation: a new atom, randomly generated in a sphere of radius Sphere, moves to the center of the structure until the distance to the nearest atom is within the limits specified in the *BondLimits* command
PCCR	*Plane-Cut Crossover:* Two randomly selected structures from the pool are cut at a random plane so that the ratio of the number of atoms in both parts of the two structures is equal, and the first parts of the two structures are swapped.
TPCR	*Two-Point Crossover:* The Cartesian coordinates of the two structures are arranged in one-dimensional vectors and divided into two parts at a random position, which is the same for both structures. Then the first parts of the two vectors are swapped.
IMOV	*Internal Move:* A randomly selected atom of a randomly selected pool structure moves a random distance towards the center of mass of the structure.
AMOV	*Angular Move:* A randomly selected atom of a randomly selected pool structure rotates by a random angle around the center of mass of the structure.
TWST	*Twist operator:* A structure randomly selected from the pool is cut by a random plane into two parts and both parts are rotated relative to each other by a random angle around an axis passing through the center of mass and perpendicular to the plane of the cut.
HINT	*Hint operator:* the new structure is selected based on the "hint", i.e. structure provided by the user. In the case of RUNNER, a file with hint structures is placed in the *./add* directory at any time the program is running. After generating the input files, the hint files are moved to the *./usd* directory. The file formats are (1) geometry optimized server program output files; (2) text files in which hint structures are in NXYZ format with *@geo* origin marks, the end of coordinates is an empty line.

Global optimization begins with the formation phase of an initial set of structures ("initial population"). This phase is called the "Initial Phase". In the course of it, a set of structures is accumulated, on the basis of which "child structures" will be generated further. The initial population is formed, as a rule, by two operators RAND and RND1 ("random generation" and "random condensation"). The preference for one of the two operators can be changed using the Init command, specifying the operators to use and the likelihood of their use. In addition, hint

structures (HINT operator) can be used for the initial phase. The initial phase ends when the minimum population (“pool”) of structures capable of forming “descendants” is formed. The number of such structures is determined by the *MinPool* command (10 by default). The structures included in the pool are sorted according to their absolute energy increase and are stored in the **.pool_* file.

At the end of the initial phase, the Main Production Phase begins. In its course, at each working cycle of the algorithm, new structures - "descendants" are formed by applying a given set of generators to one or more randomly selected structures from the "active pool". An active pool is a part of a common pool that unites the most beneficial structures (structures with lower energies). The share of the most profitable structures from the total pool size is set by the *ActivePool* keyword (by default 0.7, i.e. 70% of all pool structures). A decrease in this number can lead to an acceleration of optimization if the GM is close in structure to the structures already found. At the same time, this can lead to a loss of “biodiversity” and failure in the search for a GM of unusual structure. Excessive growth of *ActivePool* can result in slow optimization speed.

In the case when, in the course of optimization, structures appear that are similar in structure and energy to the structures included in the pool, only the one with a lower energy remains in the pool. The similarity of structures is determined by an algorithm based on the analysis of interatomic distances sorted in increasing order in both structures. Structures are considered similar if all such distances in the two structures differ by less than 5%. If, during the optimization process, the number of different optimized structures exceeds the maximum size of the pool (it is set by the *MaxPool* keyword), the new structure takes its place in the order of sorted energies in ascending order, and structures whose numbers exceed the size of the pool leave it.

Leaving a structure out of the pool does not mean that it is not used for further work. When generating a new structure, the program checks the similarity of the new structure with all structures for which the energy has ever been calculated in the current run. These structures are stored in the **.base_* file. This file contains not only the optimized structures, but also all the structures found in the intermediate local optimization loops executed by the server program. Before starting the energy calculation (local optimization) of a new structure, it is checked for similarity with all structures stored in **.base_* and the calculation starts only if no similar structures are found. Otherwise, the new structure is discarded and regenerated. In addition, the structure is discarded if its bond lengths exceed the limits defined by the *BondLimits* keyword. The number of generation attempts is determined by the *MaxTry* keyword (by default 5000). If during the specified number of attempts this generator could not create a new structure, the generation is performed by the RAND statement. In addition to the **.base_* file, RUNNER creates its shortened version, the ** .sbas_* file. It contains only the initial (generated) and final (optimized or under-optimized) structures of each calculation; intermediate optimization cycles are omitted. The **.sbas_* file is easier to parse than **.base_*. In **.base_* and **.sbas_* files, initial structures are marked with **gen*, final (in this file) - with **end*, optimized - with **opt*. In addition, when calculating the frequencies, the structure is marked with **freq*, and if there are no imaginary frequencies, **lm*. If the structure has *n* imaginary frequencies, it is marked with **nneg = <n>*.

The generated structures are written to the input data file of the server program, the shape of which is set by the *FilPat* keyword. The new input data file is placed in the *./inp* directory. From this directory, the control script (or the RUNNER program itself in the case of direct call mode) sends the input file to the server program and moves it to the ./out directory. Upon completion of the calculation by the server, the control script places the results file in the same directory. Finding the results file in the *./out* directory, RUNNER analyzes the results, extracts the structures of each optimization step and their energies from it, and writes them to the **.base_* and * *.sbas_* files. The pool and its **.pool_* file are simultaneously updated, and the input and output files from the *./out* directory are moved to the *./arc* directory (archive of all performed optimizations). Saving this data in the *./arc* directory can be canceled with the *NoSaveArchive* directive. This saves disk space, but if optimization errors occur, it will be impossible to determine their cause.

The RUNNER program terminates in several cases:

1. Local optimization of all structures, the number of which is set by the keyword *Nstruct*, is completed*;*
2. The number of program work cycles specified by the command *MaxCyc* is exceeded;
3. File *runner_stop* or *runner_prog_stop* appeared in the program directory.

The control script terminates in several cases:

1. The number of the control script work cycles specified within the script body by the parameter *maxcyc* (all characters are lower case!) is exceeded;
2. File *runner_stop* или *runner_shell_stop* appeared in the program directory.

The presence of *runner_stop* files simultaneously stops the program and the CS, the presence of *runner_prog_stop* or *runner_shell_stop* - only the program or only the CS. The content of all these files is irrelevant, they can be empty. If these files are present in the startup directory, the operation of the program or the operating system is impossible. To get started, these files must be deleted.

If the work of the program or the control system is terminated by mistake before the completion of the global optimization (GO), the restart is carried out by a command similar to the usual start. In this case, RUNNER automatically detects its previous state and continues GO. The CS acts in a similar way. A prerequisite for a successful restart is to keep intact all files in the directories *./inp*, *./out*, *./sub* as well as the **.info_* file in the startup directory. It is recommended to save copies of the **.info_*, **.out* and *res.log* files before restarting. The **.info_* file stores information about the current state of the optimization being carried out; its manual editing is not recommended.

If it is necessary to continue the completed GO (GO, in which *Nstruct* structures are examined), you must use the *Continue* directive in the **.inp* file. In this case, the current value of *Nstruct* determines the additional number of structures to be examined.

Author, licenses and reviews

The author of the program:

Prof. Stanislav K. Ignatov

Theoretical Chemistry Group

N.I. Lobachevsky State University of Nizhny Novgorod,

skignatov@gmail.com , ignatov@ichem.unn.ru , http://qchem.unn.ru

Nizhny Novgorod, Russia, 2020-2021.

Questions about working with the program, messages about possible inaccuracies and errors, feedback and wishes should be sent to skignatov@gmail.com