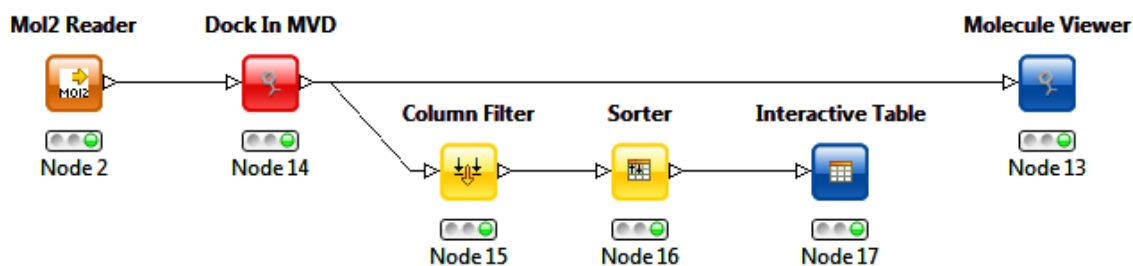


Molegro KNIME Extensions

Quick Start Guide

for Molegro Virtual Docker and Molegro Data Modeller



Molegro ApS

Copyright © 2005–2012 Molegro ApS. All rights reserved.

Molegro Virtual Docker (MVD), Molegro Data Modeller (MDM), Molegro Virtual Grid (MVG), and MolDock are trademarks of Molegro ApS.

All the other trademarks mentioned in this user manual are the property of their respective owners.

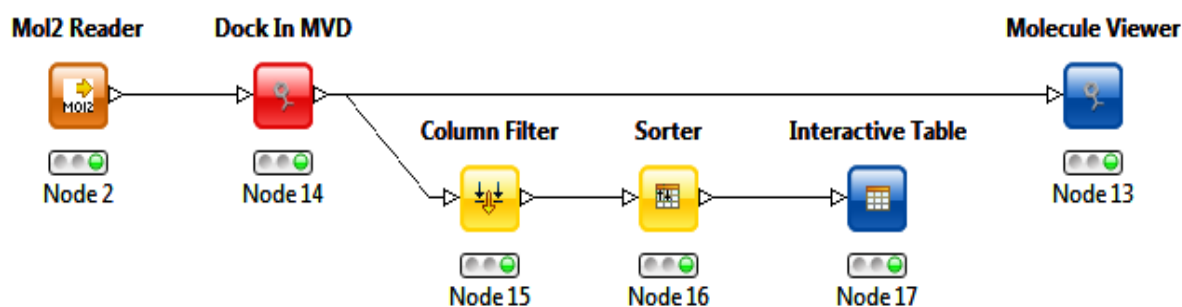
All trademarks are acknowledged.

Information in this document is subject to change without notice and is provided “as is” with no warranty. Molegro ApS makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Molegro ApS, shall not be liable for errors contained herein or for any direct, indirect, special, incidental, or consequential damages in connection with the use of this material.

Table of Contents

1 Using MVD with KNIME.....	3
1.1 Step one: preparing a workspace in Molegro Virtual Docker.....	3
1.2 Step two: setting up a KNIME workflow.....	5
1.3 Step three: analyzing and working with data in KNIME.....	7
1.4 Setting up a GPU docking run.....	8
1.5 Advanced: Setting up a parallel workflow in KNIME.....	9
1.6 Problems and Tips	12
Memory Problems.....	12
Executable locations.....	12
2 Using MDM with KNIME.....	14
2.1 Creating a Model from a Training Data Set.....	14
2.2 Using MDM Models in KNIME Workflows.....	15

1 Using MVD with KNIME



This tutorial will demonstrate how to set up a docking run with Molegro Virtual Docker in KNIME.

The tutorial assumes that you already have installed the Molegro Extensions for KNIME. Otherwise, see the Installation and Usage Guide at www.molegro.com/knime.

1.1 Step one: preparing a workspace in Molegro Virtual Docker

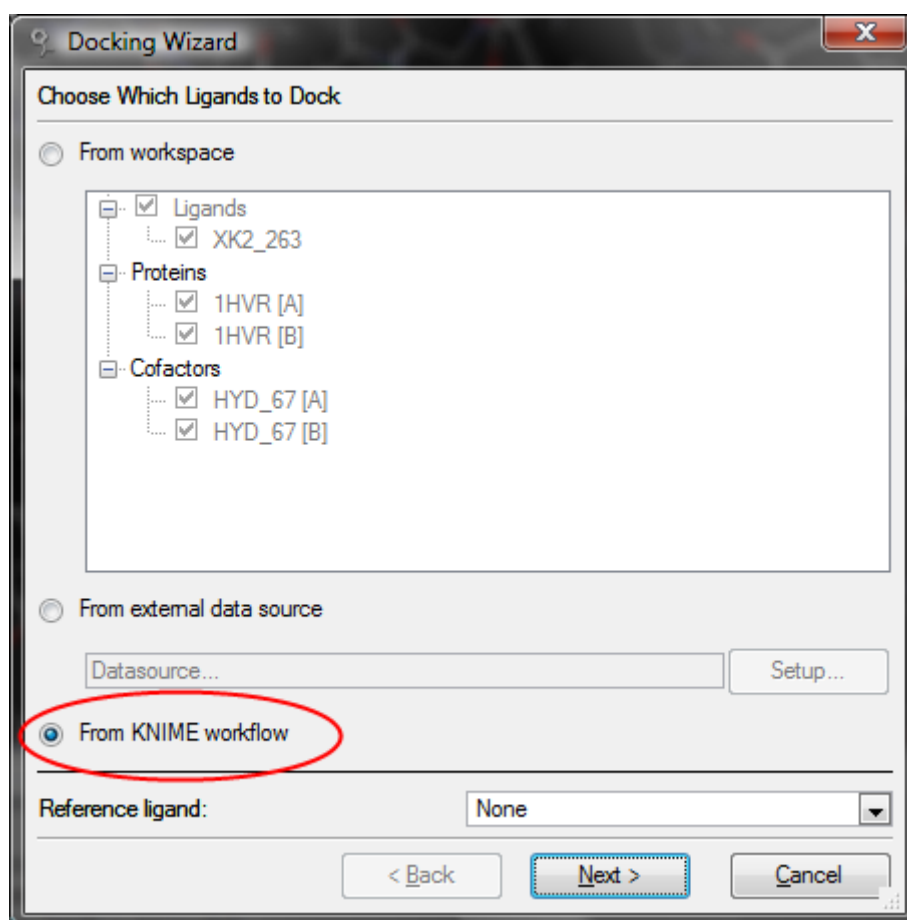
In order to use KNIME together with Molegro Virtual Docker, the first step is to prepare a workspace and a docking script. This requires the following steps:

- Import and prepare a protein in MVD.
- Do a cavity detection search to locate potential binding sites.
- Use the Docking Wizard to create a docking script for a KNIME workflow.

(Instruction on how to perform these steps can be found in the Molegro Virtual Docker user manual.)

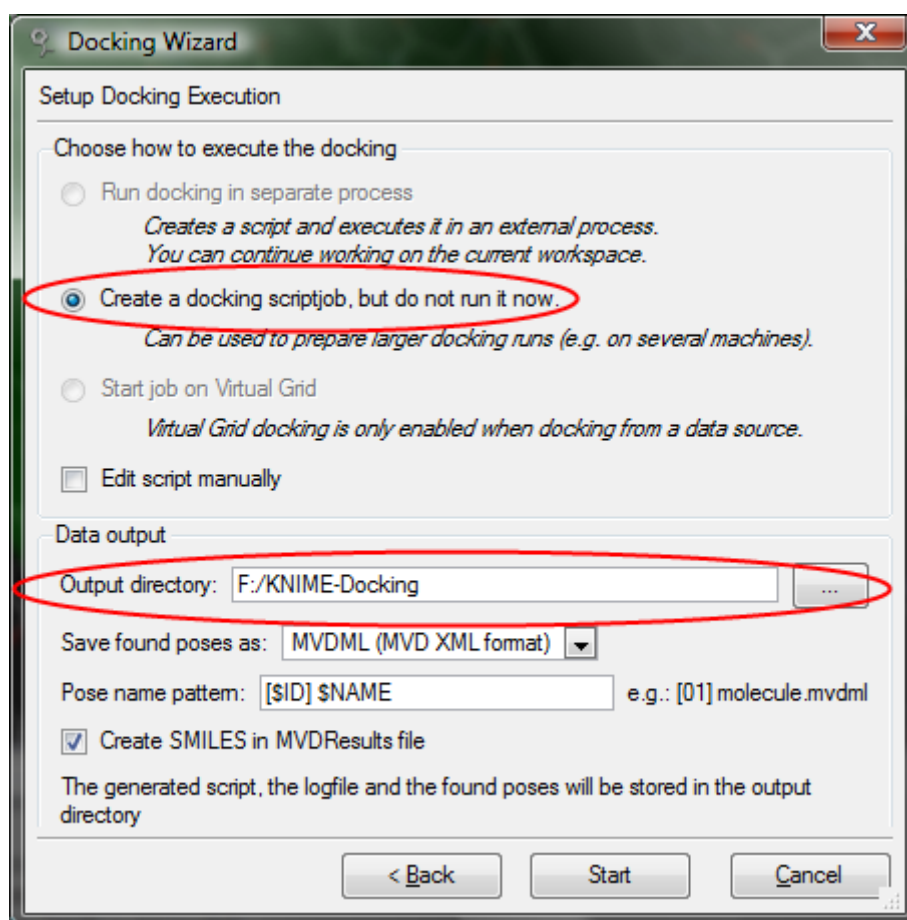
Any scripts created by the Docking Wizard can be used – the KNIME node will scan the script for ligands being docked (the 'DOCK' line), and replace those with input ligands from the KNIME workflow.

It is also possible to create a script specifically for a KNIME workflow directly from the Docking Wizard: On the first page in the Docking Wizard, choose the 'From KNIME workflow' option. The docking script will then assume that KNIME will provide the compounds to be docked.



Afterwards, set up the docking options and parameters as desired. All the functionality available in the Docking Wizard can be used in a KNIME workflow.

If you are doing virtual screening runs, it is recommended to only return one pose per compound. This will make filtering easier, and there is no need for multiple poses, if you do not plan to inspect them manually. So on the Pose Clustering tab, we recommend using the 'Virtual Screening mode' with the desired percentage set – set the percentage to 100%, if you want to test different rankings schemes.



Finally, on the Setup Docking Execution page, specify where the workspace and docking script will be saved (the 'Output directory'). Remember this location - it must be specified in the KNIME node when setting up the KNIME workflow.

1.2 Step two: setting up a KNIME workflow.

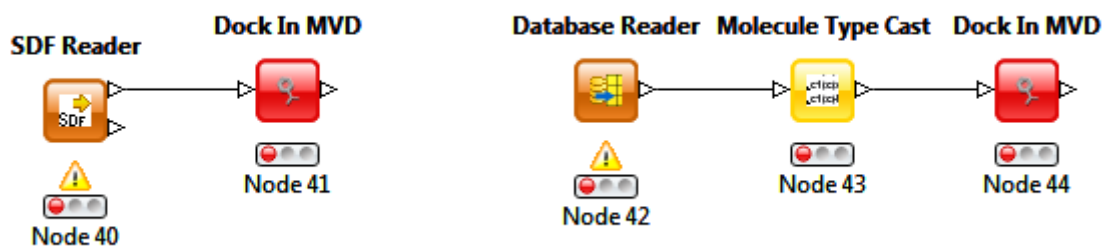
You need to create a data source in KNIME with the compounds that should be processed.

The KNIME Base Chemistry Types & Nodes contains readers for Mol, Mol2, and SDF files. These nodes will recognize and split multi-molecule files into table entries, but will not perform any preparation or validation of the chemical structures.

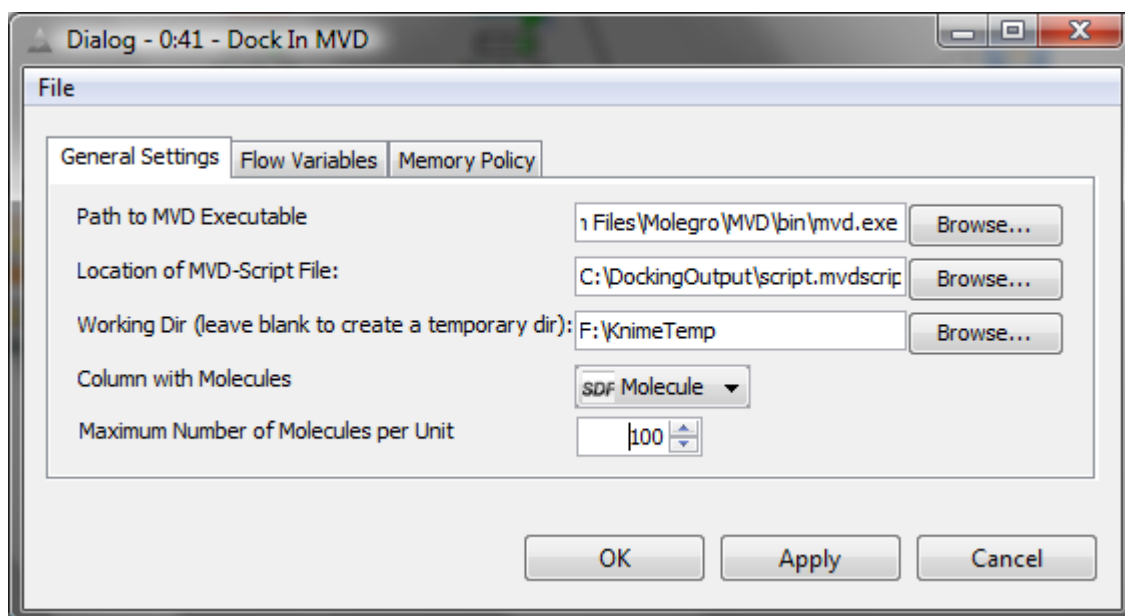
Molegro Virtual Docker reads Mol, Mol2, and SDF files natively so no conversion from the KNIME nodes is required. The structure files needs to be proper 3D structures: 2D SDF files, or SMILES strings can not be processed by Molegro Virtual Docker.

It is also possible to read molecular structures directly from a database: for example, the Database Reader node can be used to read SQL records into a KNIME table.

When reading structures from a database, KNIME can not automatically detect the type. It is therefore necessary to use the **Chemistry | Translators | Molecule Type Cast** node before molecules from a database can be used together with the Dock In MVD node.



After a data source has been set up, its output can be routed to a **Dock In MVD** node. In order to setup the **Dock In MVD** node, you need to open its settings, and specify the path to the Molegro Virtual Docker installation. Notice, that a valid license file must be present in the same directory as the MVD executable in order to run MVD.



Use the **Location of MVD-Script File** to specify where the MVD script saved by the Docking Wizard is stored.

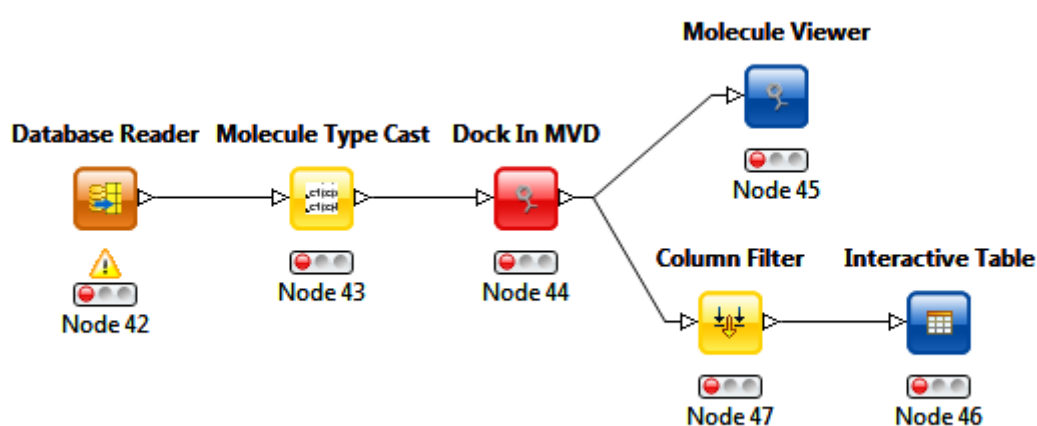
The **Working Dir** specifies where the docking poses and output files generated by Molegro Virtual Docker are saved. If no directory is specified, a temporary directory will be used - but be careful, this means that all molecule poses are lost when KNIME is closed. Notice, that even though the tables in KNIME always are stored on disk between sessions, the output table from the Dock In MVD node does not store the molecular structure in the output table: only the molecular descriptors (e.g. the different scoring function terms) and the path to the output structure is stored. So to keep the molecule poses between sessions be sure to specify a permanent working directory.

The **Column with Molecules** combobox must be set to the column in the table which contains the molecular data - notice the type icon next to the column name: it should indicate a molecular

data type like SDF, Mol2, or Mol. If the column is a text string, it will be interpreted as a filename pointing to a molecule file, so make sure the column is of the expected type.

Maximum Number of Molecules per Unit: This number controls how many molecules Molegro Virtual Docker processes for each thread that is started. Since there is an overhead associated with starting a new process (the energy grids must be precalculated), running multiple molecules per unit speeds up the docking. On the other hand, if problems are encountered (like unparseable molecules), the workflow can only resume fully completed units. The recommended setting is 10-1000 molecules.

1.3 Step three: analyzing and working with data in KNIME.



When the **Dock In MVD** node is executed, it will output a table with information about docking scores and various other energy terms. It is possible to reduce the columns presented by inserting a column filter.

Typically, you would keep at least the following columns:

Filename: The path to the molecular structure. Necessary to view the files in MVD.

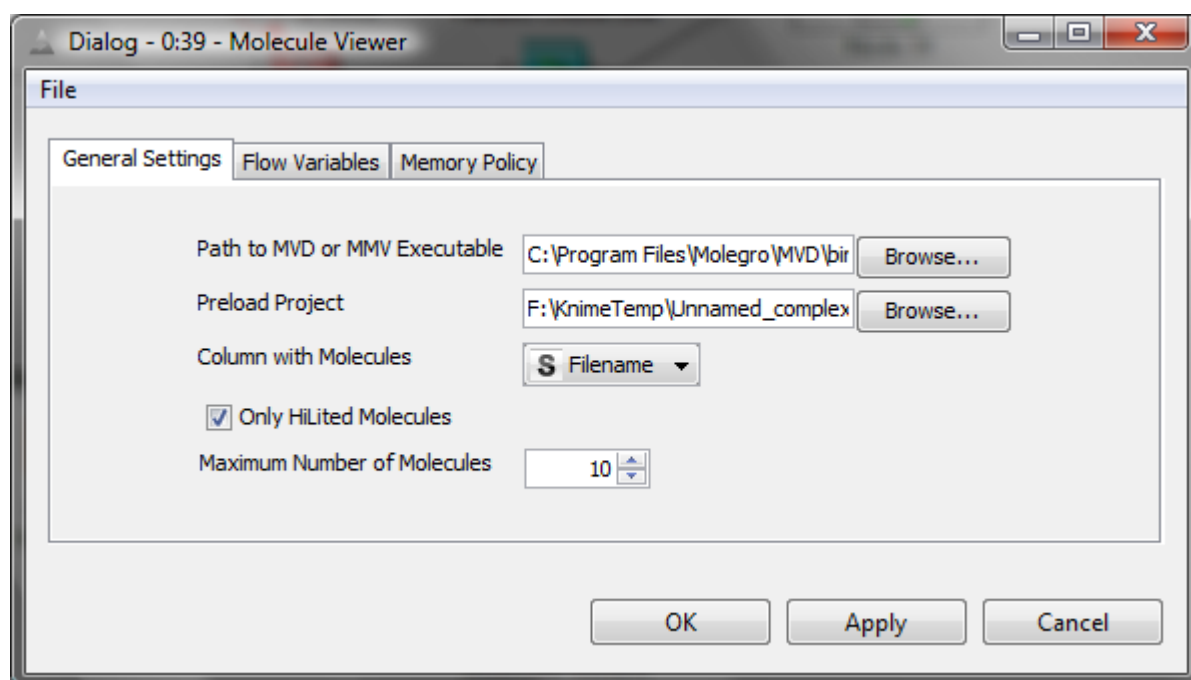
Ligand: The name of the compound being docked.

PoseEnergy: The energy assigned to the pose by the scoring function used during the docking simulation.

Energy: The MolDock energy, calculated using a non-grid scoring function.

RerankScore: A weighted scoring function mixing various descriptors with energy terms.

After having filtered the columns it is possible to sort the compounds according to one of the energy scores and select the highest ranked poses for further examination. This may be done by 'highlighting' the top ranked poses, and opening the table in the **'Molegro Extensions | Molecule Viewer'**.



The **Molecule Viewer** node is easy to set up: choose the path to the executable (either Molegro Virtual Docker or Molegro Molecular Viewer) and the column containing the molecule filenames. Notice, that since the molecular structures are not contained in the output table, we specify the string column with filenames instead. It is possible to specify a project which is loaded before the molecules: use this to load the target receptor used during the docking simulation into the molecular viewer. Finally, make sure the 'Only HiLited Molecules' is checked to only view the molecules hilited in the table.

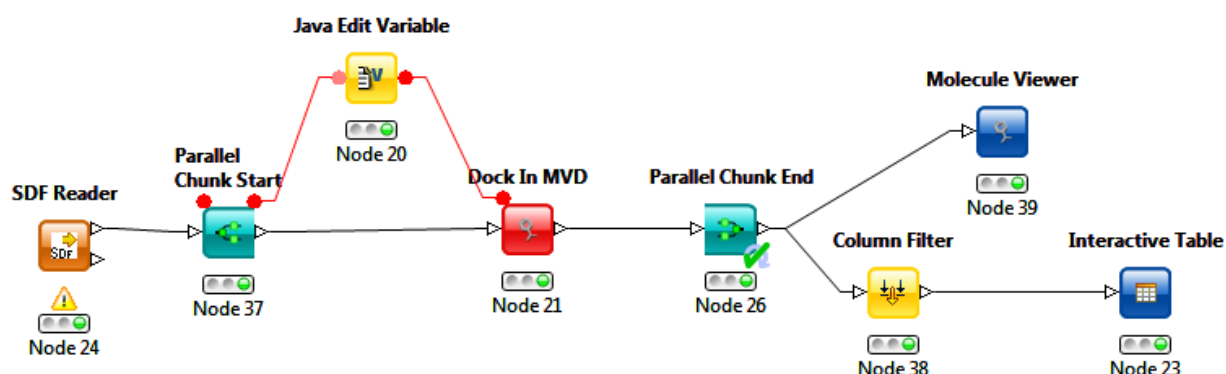
1.4 Setting up a GPU docking run.

If a Nvidia CUDA enabled graphics card is available, it is easy to setup a GPU docking run in KNIME. The KNIME node can execute any script generated by the Docking Wizard, so just select the "GPU Screening (CUDA)" search algorithm on the Customize Search Algorithm tab.

One thing to notice, is that for a powerful GPU, one single MVD CPU process might not be enough to feed the GPU with data. Therefore, it makes sense to run multiple instances of MVD even for GPU docking runs. This is explained in the next section.

It is not easy to predict the optimal number of processes for a GPU docking run. Typically, the number of physical cores on the CPU is a good choice. Too many processes might result in the graphics card running out of memory.

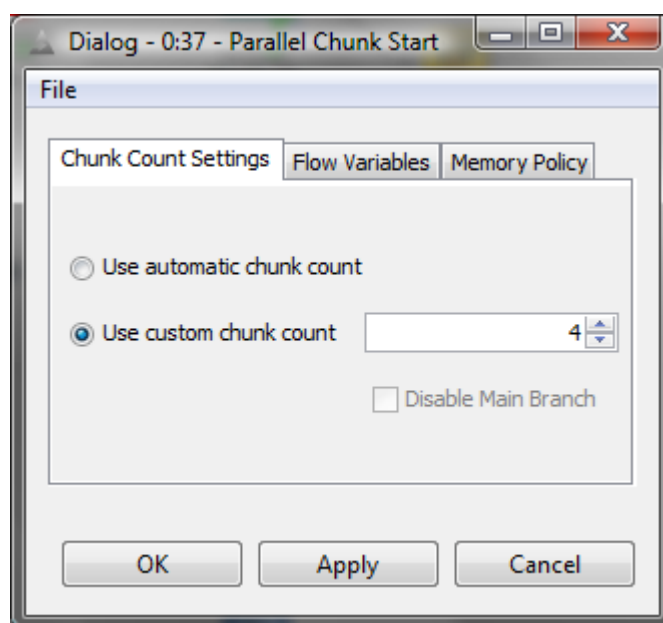
1.5 Advanced: Setting up a parallel workflow in KNIME.



Docking in MVD is single-threaded. This means that only one CPU core is used during the docking simulation. In order to better utilize the resources, it is possible to run multiple instances of MVD in parallel. As mentioned in the previous section, this also applies to GPU docking runs.

KNIME supports parallelizing workflows, but only as a *KNIME labs* feature. To set up parallelization support install the KNIME Virtual Nodes from the KNIME Labs Extensions.

As shown in the workflow above, the **Dock In MVD** node must be surrounded by a **Parallel Chunk Start** and a **Parallel Chunk End** node. Modify the number of parallel chunks using the properties for the Parallel Chunk Node:



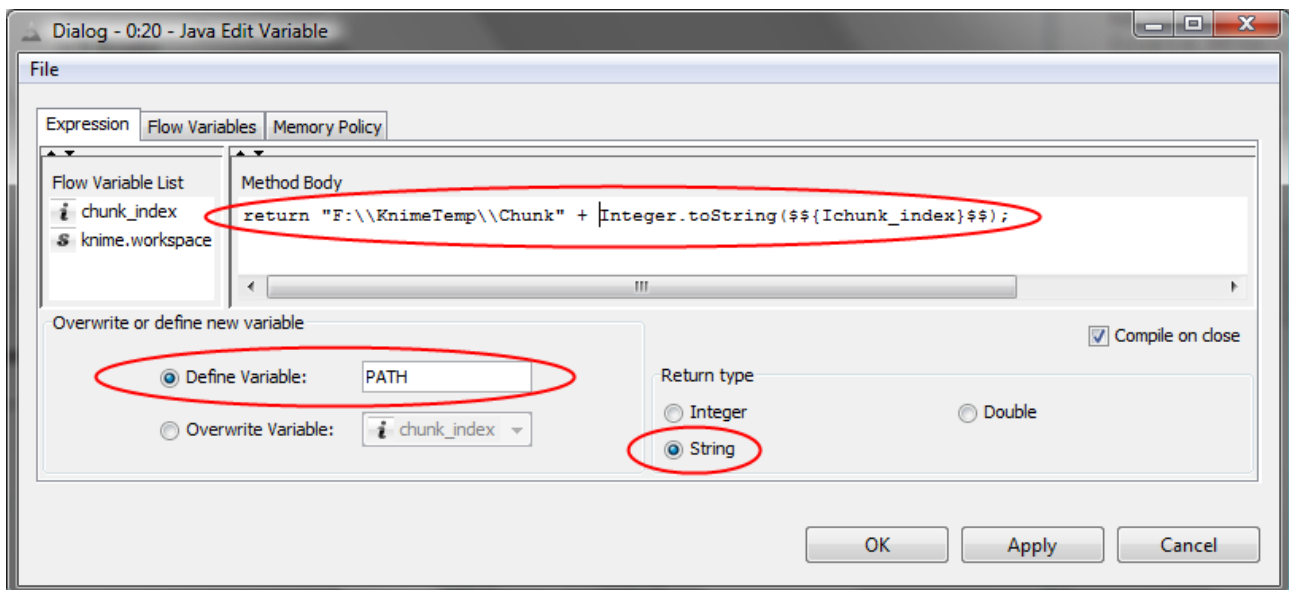
Choose a number corresponding to the number of physical cores in the machine. (The automatic chunk count in KNIME is determined as 1.5 times the number of virtual cores – e.g. 12 for a quad-

core machine with hyper-threading. This is not optimal since the processes compete on shared resources like memory and disk access.)

It is important to understand that each instance of MVD must have its own working directory – otherwise each instance overwrite the files of other instances. In order to set up this we will use flow variables to control the working directory of MVD.

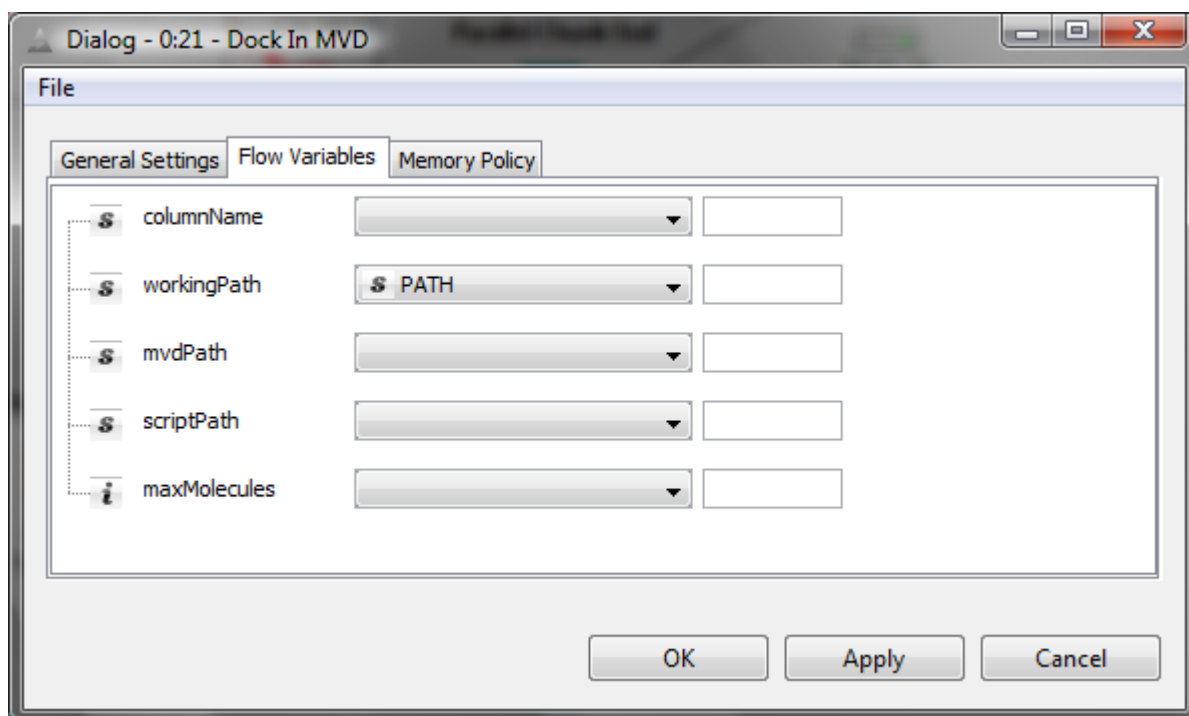
First make the flow variables port visible, by using the context menu and choose: 'Show Flow Variable Ports'.

Now we need to insert a **Flow Control | Java Edit Variable** node to construct the path file name. We will use the chunk index from the Parallel Chunk Start node to construct a new string called **PATH**, containing the file name.

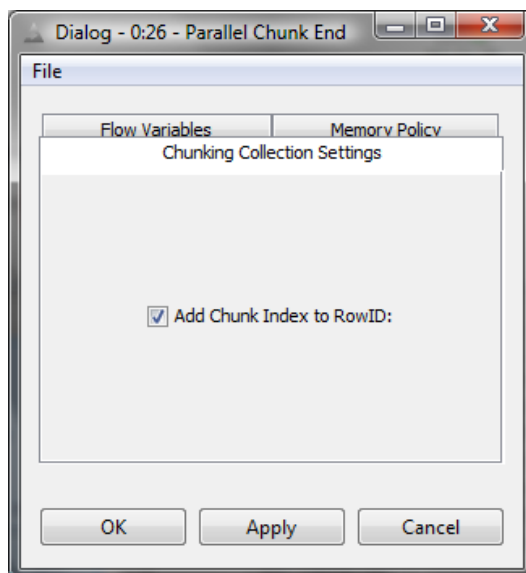


Make sure the variable name is set to PATH as in the example above, and that the return type is String. Construct a method body as above (it is possible to insert the chunk_index variable by double clicking on it). Notice that backslashes must be escaped (doubled) inside Java string literals.

Once the file name has been constructed, configure the Dock In MVD node to use the PATH as its working path (see the Flow Variables tab in the properties window).



Finally, KNIME needs a unique identifier for each row in a table. Normally, the auto-generated row ID is sufficient, but this results in ID clashes, when using the Parallel Chunk nodes. To avoid this, open the properties dialog for the Parallel Chunk End node, and make sure the 'Add Chunk Index to RowID:' is checked.



Now it is possible to execute the Dock In MVD node. Notice that for each instance that is spawned, you have to confirm the dialog message boxes that appear.

1.6 Problems and Tips

Memory Problems

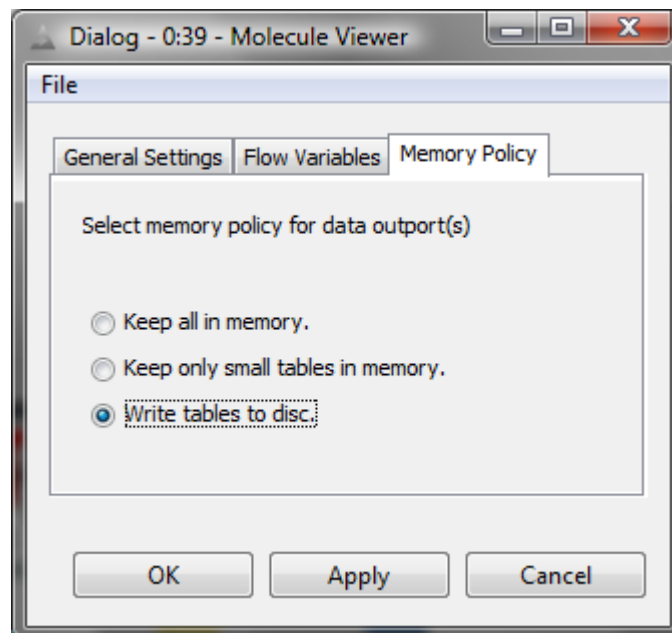
When working with large data sets, it is possible to encounter memory problems in KNIME. Typical error messages are:

Execute failed: GC overhead limit exceeded

or

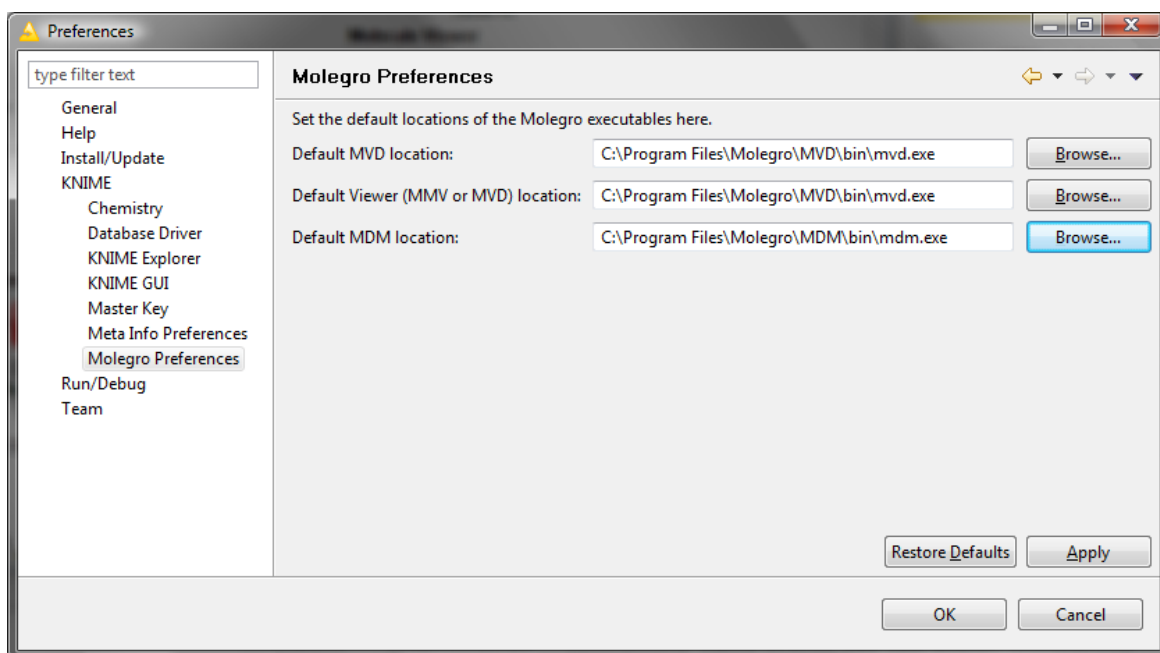
Execute failed: Java heap space

In order to prevent this, change the node properties so that tables are always written to disk:



Executable locations

The Molegro nodes need to know the location of the Molegro software executable files. Instead of specifying this every time a node is inserted, it is possible just to specify the locations once in the preferences. Choose **File | Preferences | Molegro Preferences** and set the paths accordingly:



2 Using MDM with KNIME

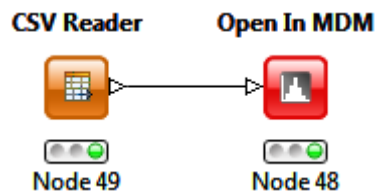
Using Molegro Data Modeller together with KNIME is normally a two-step process:

First, use KNIME to pipe a training data set into Molegro Data Modeller using the **Open In MDM** node, and train a regression or classification model using the graphical user interface.

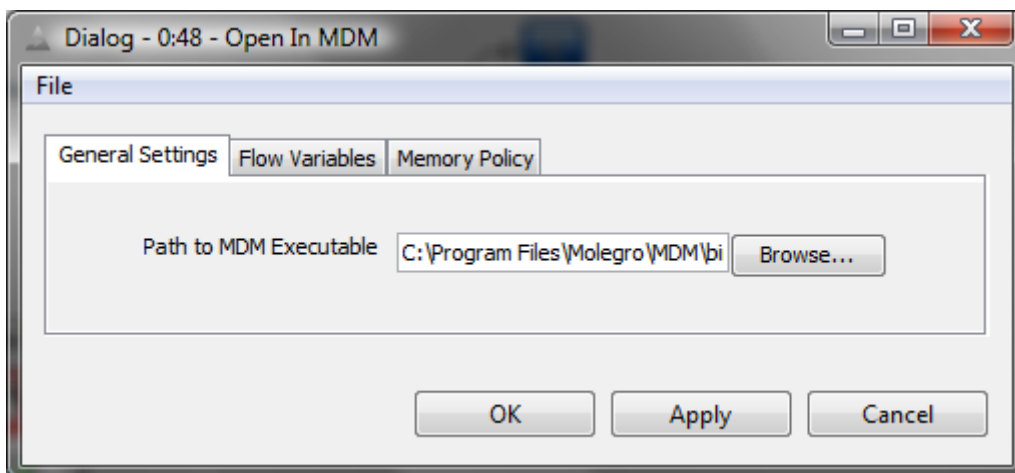
Next, save the model in Molegro Data Modeller, and use the **Apply MDM Model** node to make predictions on new data sets.

2.1 Creating a Model from a Training Data Set

It is not strictly necessary to use KNIME when building the model. Any model created by Molegro Data Modeller can be used with the **Apply MDM Model** node. However, it is important to make sure that column names are correctly matched when doing so. The easiest way to assure this, is by piping data from KNIME into the **open in MDM** node. Any KNIME table can be piped to the **Open in MDM** node, but typically a database connector or a CSV reader node is used.

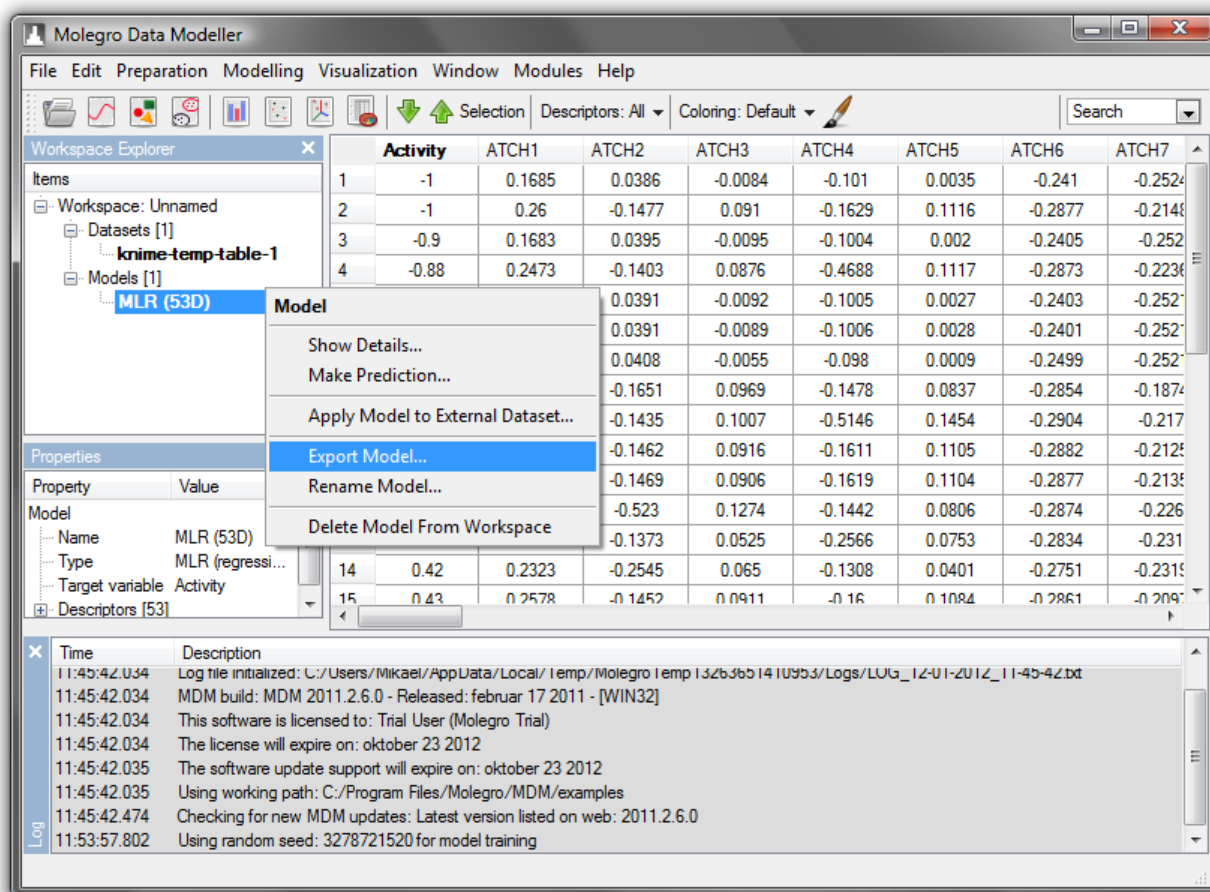


The **Open In MDM** node is easy to configure – it is only necessary to specify the location of the Molegro Data Modeller executable:



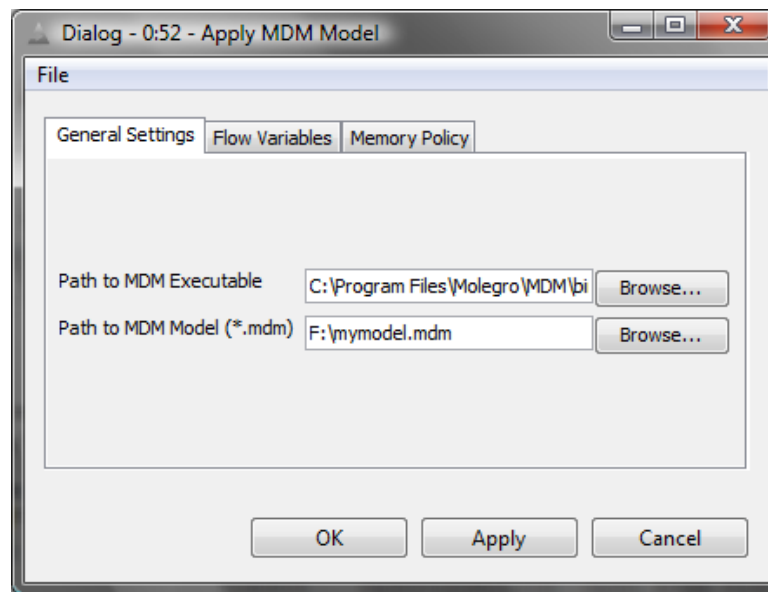
After the data has been imported into Molegro Data Modeller, a regression or classification model can be created the usual way. One of the advantages of using the Molegro Data Modeller GUI is, that it is easy to set up techniques like cross-validation, parameter tuning, and feature selection when building the model. After the model has been built, it can be easily applied to other data sets in KNIME.

To save the created model in Molegro Data Modeller, use the context menu on the generated model in the workspace (or use the **File | Export Models...** menu entry):

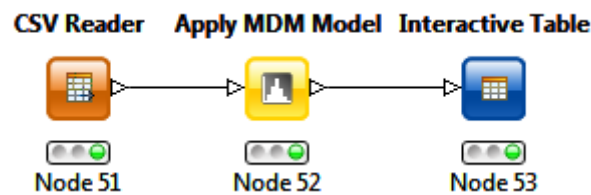


2.2 Using MDM Models in KNIME Workflows

In order to use a MDM model in a workflow, insert an **Apply MDM Model** node, and configure it to point to the generated model and the MDM executable:



Afterwards, it is easy to apply the model in a workflow:



The **Apply MDM Model** node works the same way as the **Dock In MVD** node – the input table is stored to disk, before invoking the Molegro Data Modeller process. In contrast to the MVD node, the input table is always stored to a temporary location, since all data will be stored by KNIME in the output table, when the node is finished executing.