FrEAK

Free Evolutionary Algorithm Kit

User's Guide

Project group 427

Patrick Briest, Dimo Brockhoff
Bastian Degener, Matthias Englert
Christian Gunia, Oliver Heering
Michael Leifhelm, Kai Plociennik
Heiko Röglin, Andrea Schweer
Dirk Sudholt, Stefan Tannenbaum

Thomas Jansen, Ingo Wegener

Version 0.2

February 10, 2004

Contents

List of Figures

Chapter 1

Introduction

1.1 What is FrEAK?

FrEAK, the Free Evolutionary Algorithm Kit, is a free toolkit for the creation, simulation, and analysis of evolutionary algorithms within a graphical interface.

We created FrEAK as a comfortable, flexible and extendable platform to perform experimental analysis of evolutionary algorithms. Several components commonly used in evolutionary algorithms are provided with FrEAK and a developer's guide explains how to create your own components.

The graphical interface allows you to create your own algorithms and to watch your algorithm running. With the built–in replay function, you can step back anytime and watch past runs and generations. Observers may be used to show, e.g., different performance measures, the individuals of the current population or other data derived from the algorithm.

FrEAK uses the approved random number generator "Mersenne Twister"[1] to provide well–distributed pseudorandom numbers. The random seed is drawn from the current system time on schedule creation and preserved during load and save operations.

1.2 Evolutionary Algorithms

There are many views of evolutionary algorithms in the scientific community. In FrEAK, we consider evolutionary algorithms as randomized search heuristics used to optimize a specified *fitness function*. The fitness function is defined on a *phenotype search space* which represents the set of all search points. Search points are referred to as *individuals*. An individual contains a *phenotype* used to determine the individual's fitness, a *genotype* out of the *genotype search space* representing the individual's gene data and some other attributes like, e.g., its date of birth.

[1]M. Matsumoto and T. Nishimura, "Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator", ACM Trans. on Modeling and Computer Simulation Vol. 8, No. 1, 1998, pp. 3–30.

Evolutionary algorithms in FrEAK are represented by *algorithm graphs*, also denoted as *operator graphs*. Algorithm graphs are acyclic flow graphs leading individuals through various nodes. The nodes represent *operators* like mutation, recombination, selection, and several other operators. They process the incoming individuals and propagate the resulting individuals through their outgoing edges.

Every generation, the current population[2] is led through the algorithm graph from a start node towards a finish node where the new population is received. A graphical editor is provided that enables you to design your own algorithm graphs. For further information, read Chapter 4: Designing Algorithms.

1.3 Overview

FrEAK is based on *schedules* containing the algorithm and simulation options. A wizard guides you through the process of creating a new schedule, as described in Chapter 2. You may save the current schedule to share the results of your algorithm or to continue the simulation later on.

When a schedule has been created, lean back and start the simulation. FrEAK provides a replay mode allowing you to watch past runs and past generations in the current run. To learn more about controlling runs, read Chapter 3: Controlling Runs.

1.3.1 Schedule Creation: Designing the Algorithm

The algorithm consists of seven components, which are called *modules*:

- a phenotype search space,
- a fitness function,
- a genotype-mapper,
- an algorithm graph,
- a set of stopping criteria,
- a population model, and
- an initial population.

The phenotype search space represents the domain of the fitness function. The fitness function is the function to be optimized. While most of the modules provided with FrEAK are designed for maximization problems, this is not a fundamental restriction; you may write custom modules that work with minimization problems as well.

[2]or subdivisions of it, see Section 2.4.6 on population models

The genotype-mapper is an optional component. If a genotype-mapper is specified, it is used to map an individual's phenotype to a corresponding genotype. The operators inside the operator graph then work on the individual's genotype. If no genotype-mapper is specified, the genotype search space equals the phenotype search space and the individuals's phenotype represents its genotype, too.

To specify the algorithm graph, you may create your own algorithm with the graphical editor or load an algorithm. Several common algorithms are provided with FrEAK and can be picked from a ready–made list.

Stopping criteria tell the algorithm to stop the current run if a specified condition is fulfilled. The population model is an optional choice and allows you to maintain subdivisions inside the population. The last step here is to specify the initial population by choosing an *initialization* module that creates the individuals forming the first generation. If you want to write your own module, refer to the *Module Developer's Guide*.

1.3.2 Schedule Creation: Preparing the Simulation

Preparing the simulation covers the following two steps:

- add observers and views and
- create batches of runs.

Observers are modules collecting and computing data that is then displayed by *views*. E.g., observers can compute measures like the fitness variance within the population or simply collect all individuals of the current population. A view displays the data observed by an observer if it is able to handle the type of data the observer provides. The two most commonly used data types are *individuals* and *numbers*. Observers may have an arbitrary number of views.

Last but not least, you may create *batches* of runs. A batch is a collection of runs with own configurations for the phenotype search space, the fitness function and the initialization module. The modules selected and configured in the previous steps form the first batch which is created by default. You may then eventually add other batches with different configurations, e.g., to vary the dimension of the search space or the population size.

1.3.3 Controlling Runs

When the schedule creation is finished, you see a control panel at the bottom, an information panel to the left and the created views to the right. For details, see Chapter 3: Controlling Runs.

Note that a speed limit of 10 generations per second is set by default in the control panel. You can switch off the speed limit or alter its value by using either the text field or the slider to the right.

Then hit the **Start** button to start the simulation. When the stopping criterion is fulfilled, the run ends and the next run starts, if multiple runs or batches have been created.

Please note that several options and commands are disabled when the algorithm is running. To pause the algorithm and enable editing commands, hit the **Suspend** button to suspend the current run. For further information about editing the current schedule, see Section 3.3: Editing the Current Schedule.

In the information panel, the current time index is shown. By entering a different time index or by using the navigation buttons in the control panel, you can skip back to past generations, runs, and batches to watch a replay of the simulation at that point of time. To continue the simulation, hit the **Skip to End** button and FrEAK continues creating new generations.

Chapter 2

Creating a Schedule

2.1 Some Words about Schedules

The *schedule* is the central object you deal with in FrEAK, so it is necessary to explain it a little bit more in detail.

2.1.1 What is a Schedule?

A schedule is the collection of all data that is needed to simulate one or several runs of your algorithm. It contains the algorithm you want to simulate with all its components and the preferences for the simulation itself.

2.1.2 What is a Batch?

The runs that are to be simulated consist of one or multiple *batches*. A batch is a collection of configurations for the phenotype search space, the fitness function, and the initial population. Furthermore, it contains the number of runs you want to simulate with these settings. A schedule will contain at least one batch with at least one run with the settings defined using the **Schedule Editor** but you are free to either increase the number of runs or to append new batches to your schedule.

It is important to realize that the modules of the schedule stay the same throughout all runs and batches and that a batch simply gets new configuration settings for these modules.

A batch consists of the following four components:

- the number of runs,
- configuration settings for the phenotype search space,
- configuration settings for the selected fitness function, and
- configuration settings for the initialization module that creates the initial population.

2.2 Overview and Navigation

Most certainly, you now want to create your own schedule to see some algorithms in action, so let's begin right away.

First of all, select **New** from the **File** menu:

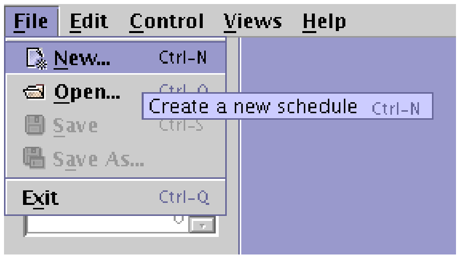
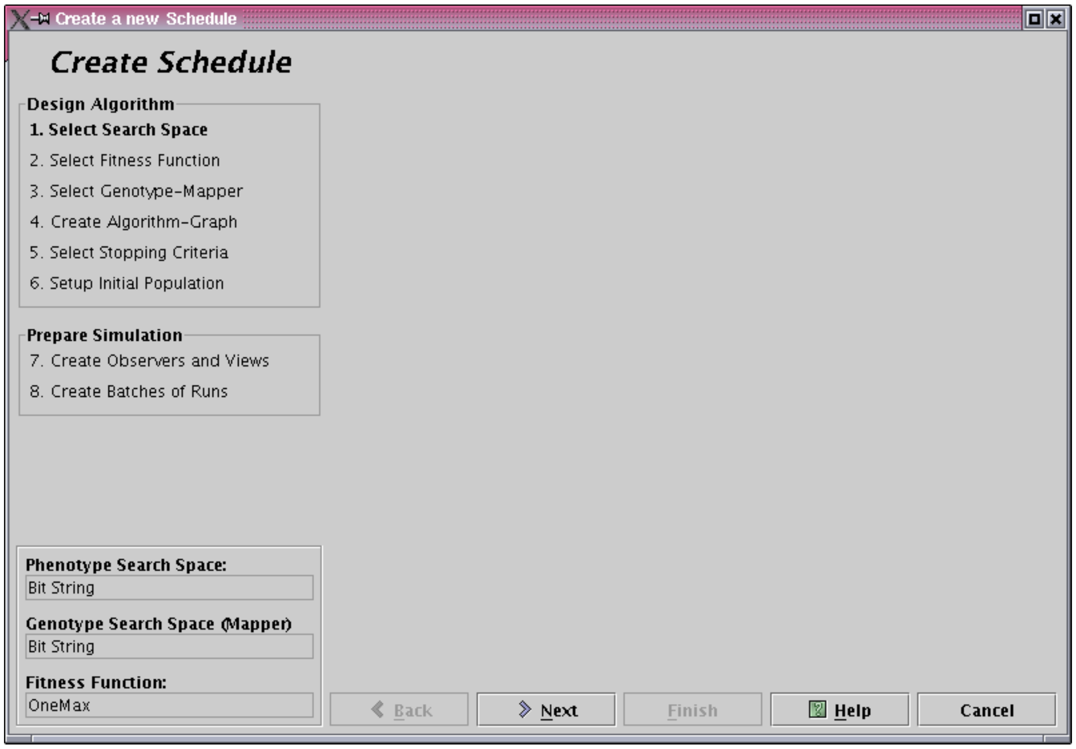


Figure 2.1: Creating a new schedule.

You will be presented the **Schedule Editor**.


Create a new Schedule
Create Schedule
Design Algorithm
1. Select Search Space
2. Select Fitness Function
3. Select Genotype-Mapper
4. Create Algorithm-Graph
5. Select Stopping Criteria
6. Setup Initial Population
Prepare Simulation
7. Create Observers and Views
8. Create Batches of Runs
Back
Next
Finish
Help
Cancel

Phenotype Search Space: Bit String
Genotype Search Space (Mapper) Bit String
Fitness Function: OneMax

Figure 2.2: The Schedule Editor.

The dialog consists of three parts. At the bottom of the window, you find the navigation buttons **Back** and **Next**. Use them to navigate back and forth through the wizard. Press **Help** to get context-sensitive help and hit **Finish** to end editing the schedule and run it. At any time you can press **Cancel** if you changed your mind and discard all changes you made to the schedule.

The progress in the wizard is displayed on the left side of the window. The step you are working on is highlighted. In the lower left corner you can see the chosen search spaces and the fitness function.

The rest of the dialog will contain panels where you can construct your schedule step by step. So let's take a quick tour through the **Schedule Editor**.

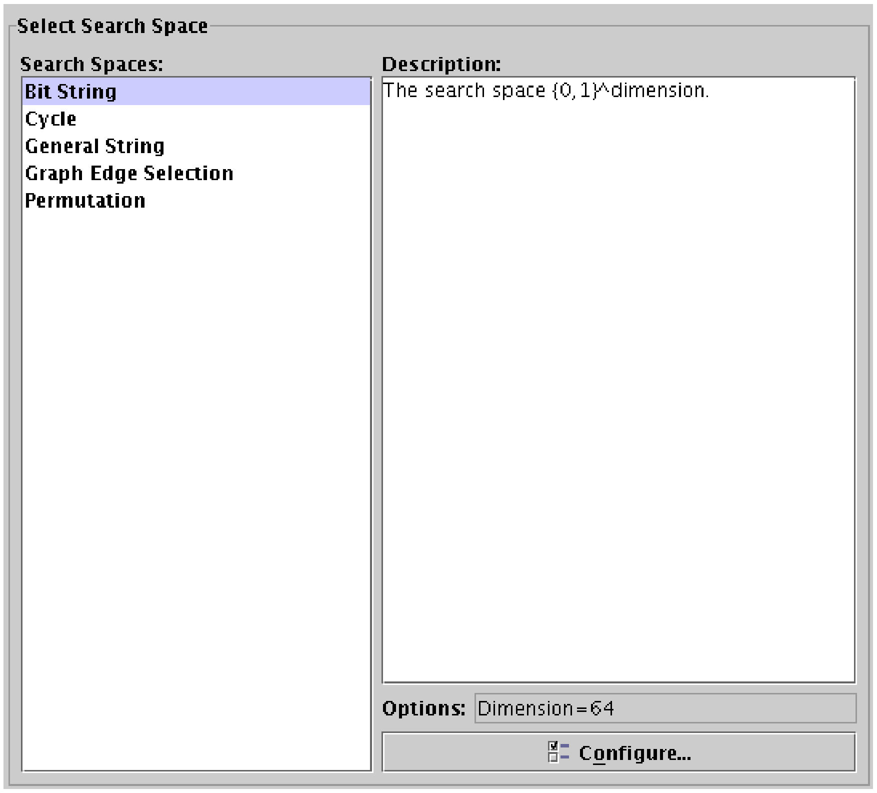
2.3 General Notes

Before you start creating a schedule keep the following in mind:

- Whenever you can select a module from a list, there is most probably a **Configure** button near it. So whatever you choose, check the configuration of the module if it suits your needs. If there is a text field above the button that says "**Options**", this field will contain the current configuration of the selection in a human readable format. For more information about configuring modules, see Section 2.6: Configuration of Modules.
- Most modules you select from a list will show their description in a text area beside or below the list.
- The navigation buttons at the bottom are context sensitive. So, if **Finish** is enabled, your schedule is syntactically correct and you **may** run it. If, for example, the **Next** button is disabled, you forgot to select or configure something in the current step.
- Several modules of the schedule depend heavily on other modules. So, when you exchange a module for another one, e.g., if you change the phenotype search space, modules selected and configured in following steps may be removed from the schedule and might even not be available for selection any more.

2.4 Designing the Algorithm

2.4.1 Step 1: Select a Search Space



The search space {0,1}^dimension.

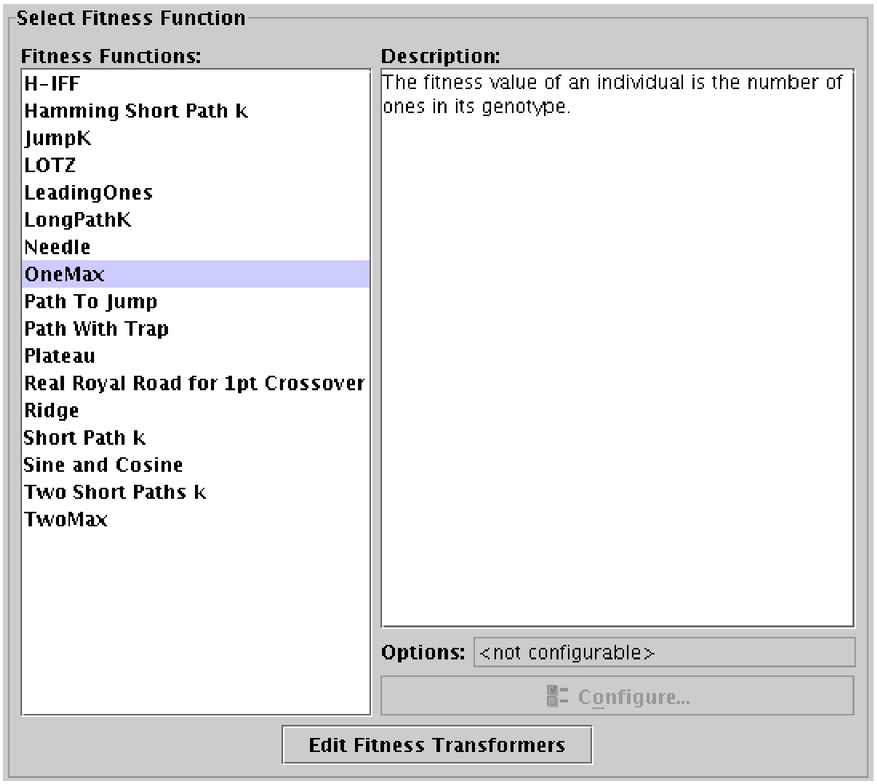
Options: Dimension=64

Figure 2.3: Selecting a search space.

First, you need to choose a *phenotype search space* as a domain for the upcoming fitness function from the list (see Figure 2.3). Bit String is the default selection right now. If you want to change the search space, just click on another one and configure it using the **Configure** button.

For now, we are satisfied with Bit String with a dimension of 64. Press **Next** to continue.

2.4.2 Step 2: Select a Fitness Function



The fitness value of an individual is the number of ones in its genotype.

Options: <not configurable>

Figure 2.4: Selecting a fitness function.

In the second step you need to choose the *fitness function* your algorithm is supposed to optimize.

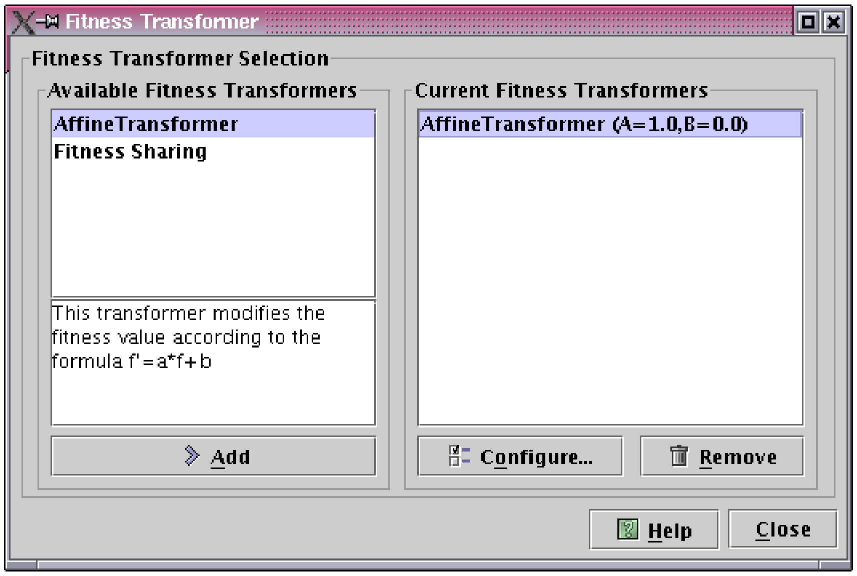
Just like the search space you can choose it by selecting one from the list and configure it by pressing the **Configure** button (see Figure 2.4).

The **Edit Fitness Transformers** button is an advanced option to define *fitness transformers* that provide additional functionality to your fitness function. Fitness transformers are applied to your fitness function and transform the computed values.

If you want to know how to use fitness transformers, read the next paragraph, otherwise simply select RIDGE as fitness function and press **Next** to continue.

Defining Fitness Transformers

If you want to add fitness transformers to your schedule, click on **Edit Fitness Transformers**. You will be presented the following dialog:


Fitness Transformer
Fitness Transformer Selection
Available Fitness Transformers
Current Fitness Transformers
Add
Configure...
Remove
Help
Close

AffineTransformer
Fitness Sharing

This transformer modifies the fitness value according to the formula f'=a*f+b

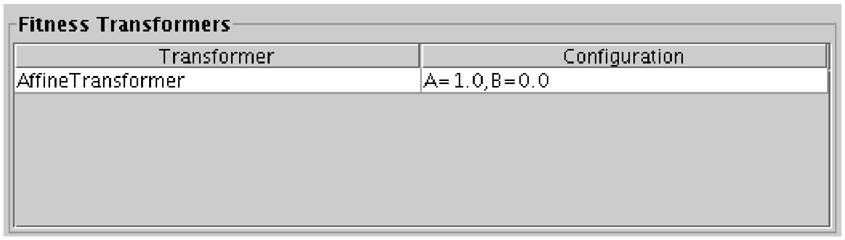
AffineTransformer (A=1.0,B=0.0)

Figure 2.5: Defining fitness transformers

On the left side you can choose the fitness transformer you want to apply on your fitness function. Click on **Add** to use the selected transformer. The transformer will then appear in the list of your **Current Fitness Transformers** on the right. You can apply as many transformers as you want, they will be applied in the order specified by the list.

Of course, you can configure each transformer after adding it to the list of **Current Fitness Transformers** by clicking on the **Configure** button.

If you are satisfied with the fitness transformer setup, click on **Close** to get back to the schedule editor dialog. You will notice a small table containing information about your selected fitness transformers below the List of available fitness functions (see Figure 2.6).



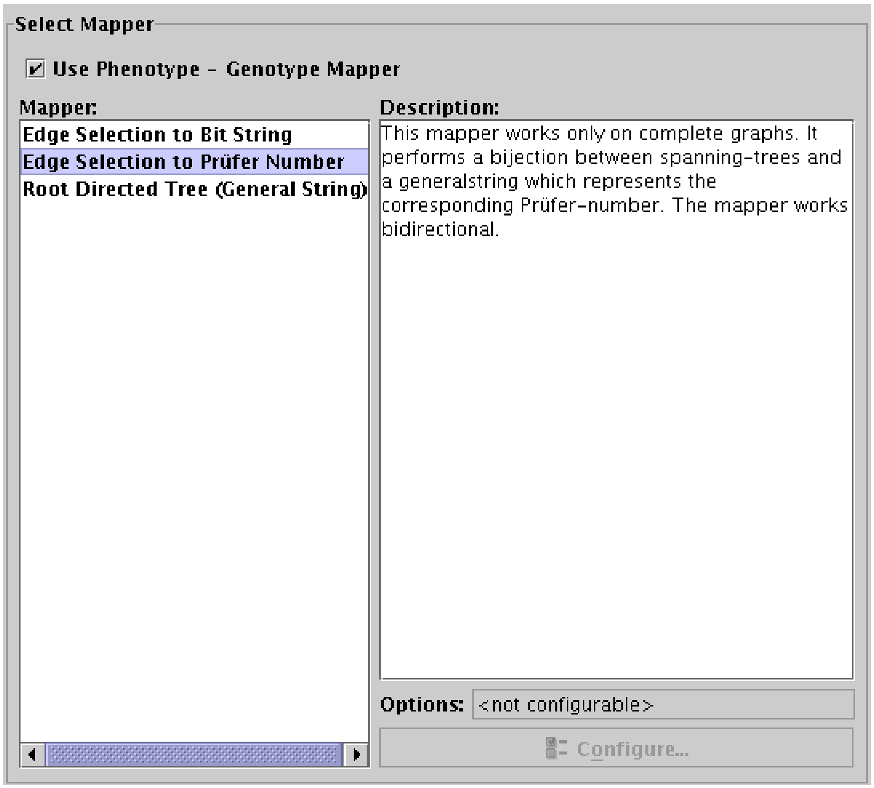
Transformer	Configuration
AffineTransformer	A=1.0,B=0.0

Figure 2.6: Setup of fitness transformers.

But let's get back to editing the schedule...

2.4.3 Step 3: Choosing a Genotype-Mapper

Some search spaces represent their individuals in a way that is inappropriate for the mutation and crossover modules you want to use or sometimes you just want to represent an individual differently from their original encoding. An obvious example is the use of the search space **Graph Edge Selection**. This search space works with an internal encoding of edge selections of a graph. Now imagine you want to represent graphs (spanning trees in this case) using a Prüfer number. You can either write a completely new search space and new mutation and crossover modules or you can *map* the individuals to a new search space, in this case to the search space **General String** using a **Mapper**. We call the the individuals that are created by the original search space **Phenotypes** and the individuals being produced by the mapper **Genotypes**. If you don't select any mapper, phenotypes and genotypes are obviously identical.


Select Mapper

☑ Use Phenotype - Genotype Mapper

Mapper:
- Edge Selection to Bit String
- Edge Selection to Prüfer Number
- Root Directed Tree (General String)

Description:
This mapper works only on complete graphs. It performs a bijection between spanning-trees and a generalstring which represents the corresponding Prüfer-number. The mapper works bidirectional.

Options: <not configurable>

Figure 2.7: Choosing a Genotype-Mapper.

Before you can select a mapper you have to mark the checkbox **Use Phenotype - Genotype Mapper**. Then you can select one of the offered mappers from the listbox and configure it the same way you did it with the search space and the fitness function. That's all you have to do.

Note:
If you selected a mapper in this step the rest of the schedule creation process assumes that your individuals are of the same type the mapper produces. So if you have the search space **Graph Edge Selection** and choose the **Edge Selection to Prüfer Number** mapper that maps the phenotypes to genotypes of type **General String**, the rest of the schedule creation wizard will work as if you had selected the **General String** search space right from the beginning.

At the moment we don't need any mappers, so just click on **Next** to advance.

2.4.4 Step 4: Create an Algorithm Graph

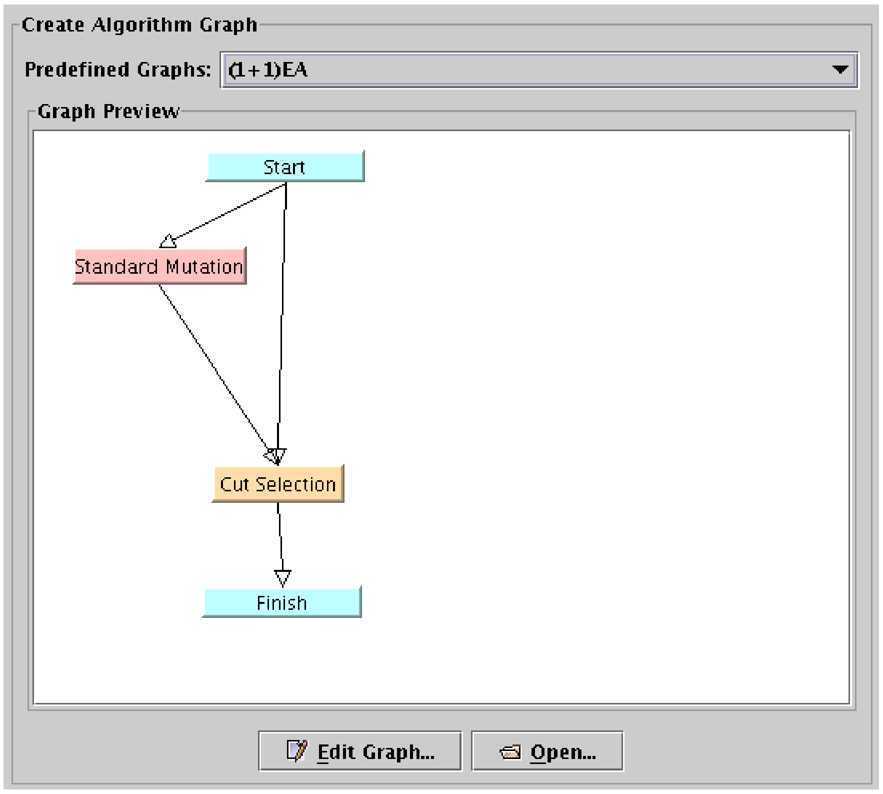


Figure 2.8: Creating an algorithm graph.

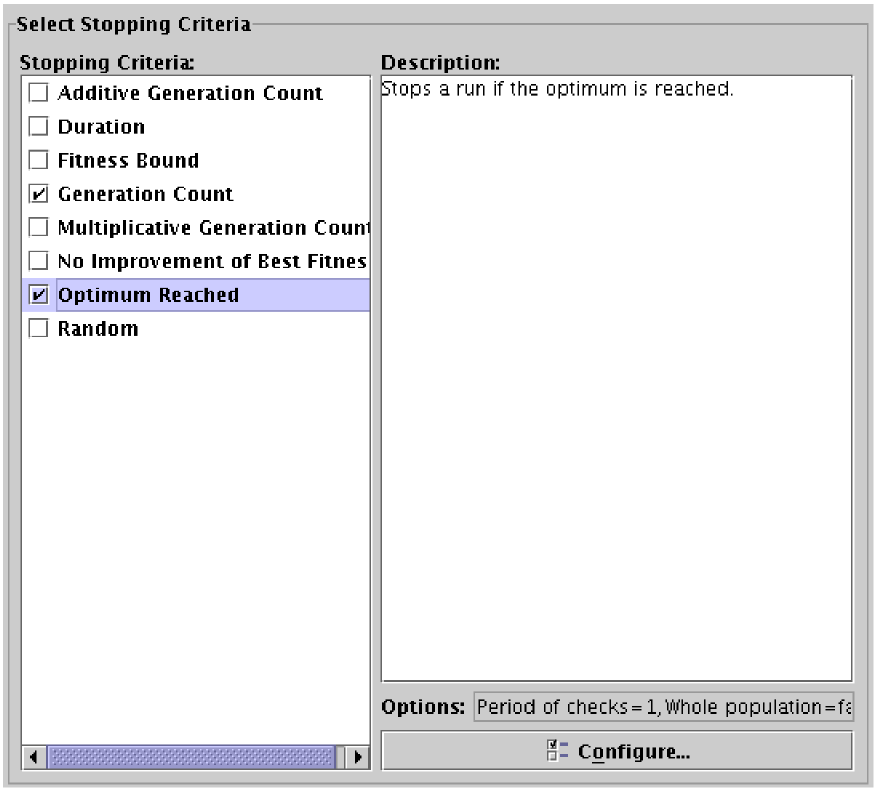
In this step you specify the *algorithm graph* which is the core of your algorithm. For further explanations on algorithm graphs, read Section 1.2: Evolutionary Algorithms or Chapter 4: Designing Algorithms.

In the center of the window, you see a preview of the current algorithm graph. You can either choose a predefined graph from the drop-down list on top, or create your own graph by clicking the button **Edit Graph**.

For now we use the default graph. To see how to create custom graphs, see Chapter 4: Designing Algorithms.

Press **Next**.

2.4.5 Step 5: Select Stopping Criteria



Stopping Criteria:

- [ ] Additive Generation Count
- [ ] Duration
- [ ] Fitness Bound
- [x] Generation Count
- [ ] Multiplicative Generation Count
- [ ] No Improvement of Best Fitnes
- [x] Optimum Reached
- [ ] Random

Description: Stops a run if the optimum is reached.

Options: Period of checks = 1, Whole population = fa

Figure 2.9: Selecting stopping criteria.

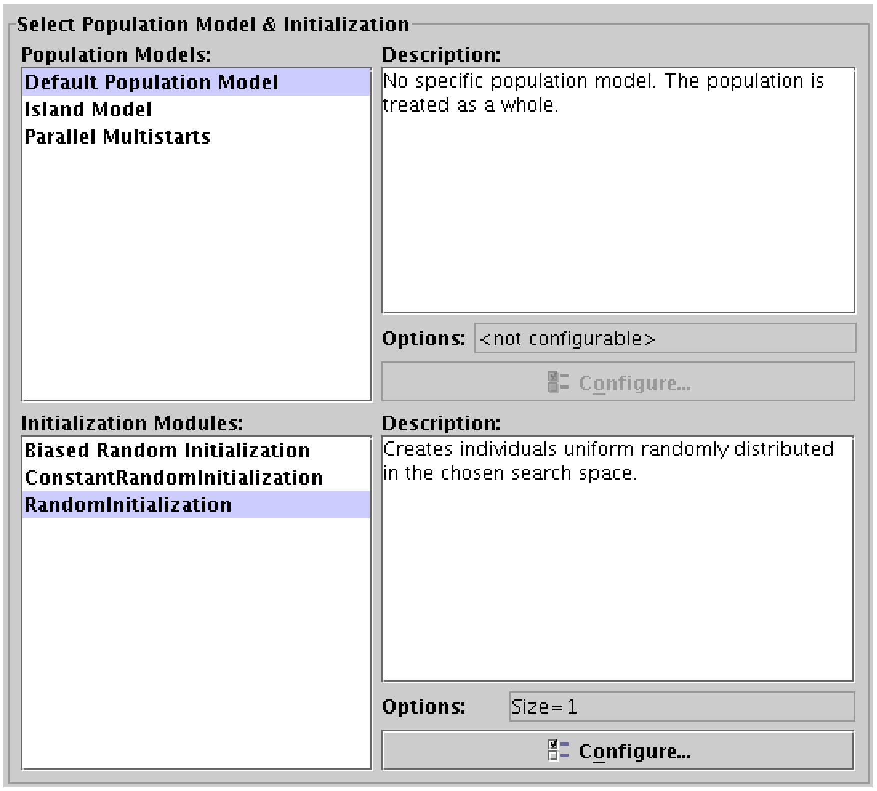
Stopping criteria tell the algorithm to stop the current run if a specified condition is fulfilled. Select one or more stopping criteria from the list of available stopping criteria (Figure 2.9). The check boxes will show you which stopping criteria have been selected. If you select multiple stopping criteria, the run is stopped if any of the criteria is fulfilled. In case no criterion is selected, you have to end the run manually by pressing the **Skip Run** button in the main window.

After selecting a stopping criterion in the list you can configure it by pressing the **Configure** button.

Select Optimum Reached and press **Next**.

Note:
The Optimum Reached stopping criterion is a good example for a module that is not available with every fitness function. Fitness functions that don't know their optimal value will not allow this stopping criterion so that it won't be displayed in the list of available stopping criteria.

2.4.6 Step 6: Select Population Model and Initialization



No specific population model. The population is treated as a whole.

Options: <not configurable>

Creates individuals uniform randomly distributed in the chosen search space.

Options: Size=1

Figure 2.10: Selecting population model and initialization.

This step consists of two parts that both deal with the population of the algorithm and how this population is initialized. In the upper panel you select your *population model*. The population model is an optional choice and allows you to maintain subdivisions inside the population. The Default Population Model is appropriate if you do not want to use subdivisions at all.

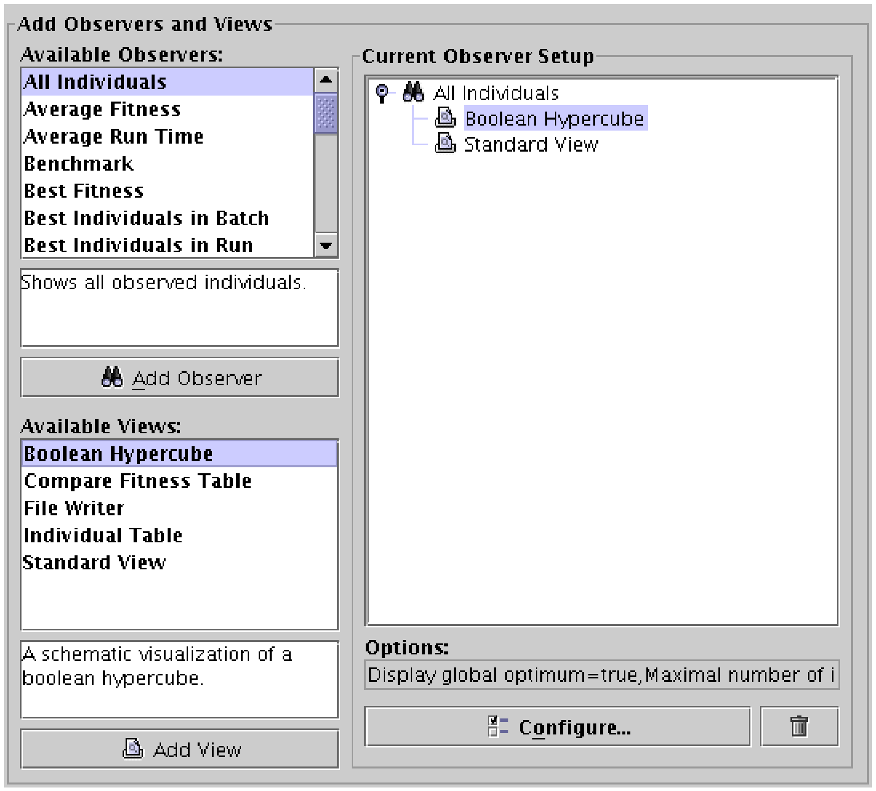
In the lower panel you select the *initialization module* which creates the initial population.

For now, leave the population model and the initialization module as it is to create a randomly chosen population of size 10.

With the completion of this step, you also completed the algorithm itself. What's missing is the collection and display of the data the algorithm produces. This is configured in the next step.

2.5 Preparing the Simulation

2.5.1 Step 7: Add Observers and Views



Shows all observed individuals.

A schematic visualization of a boolean hypercube.

Display global optimum=true,Maximal number of i

Figure 2.11: Adding observers and views.

Now that your algorithm is complete, you surely want to see what it does and how it performs. FrEAK uses two types of modules to accomplish your needs, *observers* and *views*. Observers collect and compute data that is produced by the algorithm and then displayed by one or several views on the screen or written to a file. There is (virtually) no limit of the number of observers and views you can use to process data.

The **Current Observer Setup** shows your current setup of selected observers and views in a tree view where observers are containers for views and views are leaves in the tree.

At first, the **Current Observer Setup** is empty, so you will most probably want to add an observer now. Select ALL INDIVIDUALS from the list of available observers and press the **Add Observer** button or double click on the list entry to add the observer to your current setup. This Observer just forwards the individuals to its views so they can display them in whatever way they want.

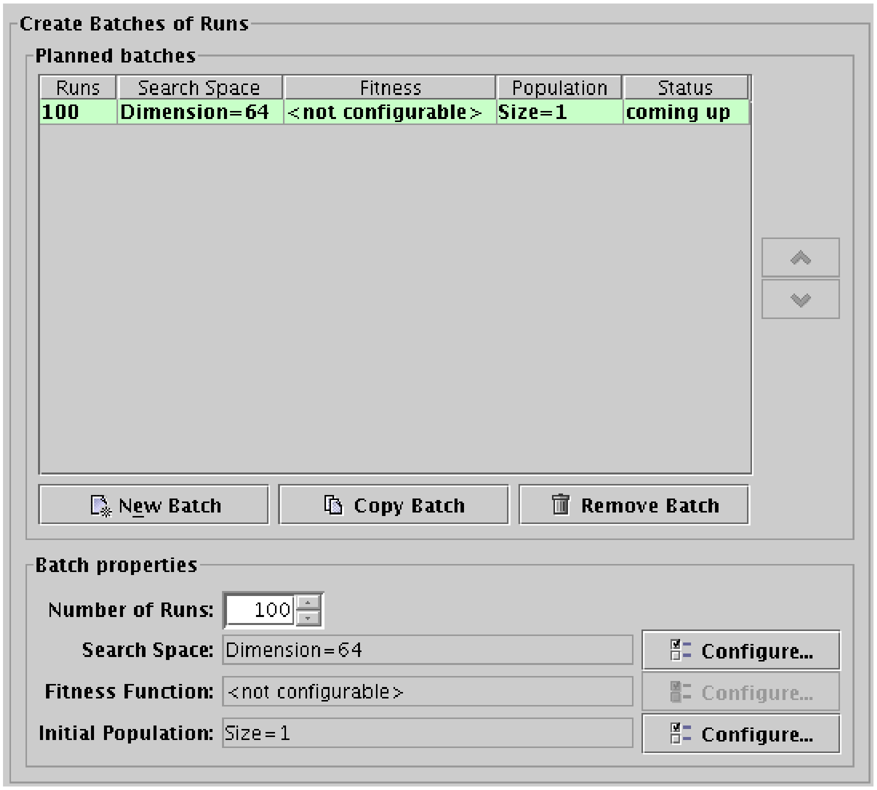
Since observers without any views do not make sense, you should add some views now, too. Select BOOLEAN HYPERCUBE from the list of available views and press **Add View** or double click on the list entry. The BOOLEAN HYPERCUBE displays BIT STRING individuals in a graphical visualization. Also add the view INDIVIDUAL TABLE to the observer. This view displays all individuals in a table.

Note:
The list of available views depends on your selection in the **Current Observer Setup**. A view has to match the observer it is assigned to so the list contains all views that match the currently selected observer (if you selected an observer) or the owning observer of the view (if you selected a view).

You can delete observers and views by selecting them and pressing the trash can button.

You should now have the setup as shown in Figure 2.11.

2.5.2 Step 8: Creating Batches of Runs



Runs	Search Space	Fitness	Population	Status
100	Dimension=64	<not configurable>	Size=1	coming up

Number of Runs: 100
Search Space: Dimension=64
Fitness Function: <not configurable>
Initial Population: Size=1

Figure 2.12: Creating batches of runs.

If you have read Section 2.1.2: What is a Batch?, you should know by now what a batch is. Here, you can setup your batches. Figure 2.12 shows the first batch that is created automatically at the first time you enter this panel or hit the **Finish** button. As you can see, the table-row has a light green background and the last cell of the row (in the status-column) says "*coming up*". Rows can have 3 different colors which reflect the progress of the corresponding batch:

green background / status: coming up The batch has not yet started. You can modify any setting of this batch.

yellow background / status: running The batch is currently running. At least one run has already started. You cannot modify anything but the number of runs this batch has. You cannot decrease the number of runs below the number of already completed runs (plus the actual running one) though.

red background / status: finished This batch has been finished completely. All runs have completed. Nothing can be changed here.

You can add new batches by either pressing **New Batch** or **Copy Batch**. The **New Batch** button creates a copy of the default batch that contains the configurations made in the first panels. The **Copy Batch** button simply copies the selected batch and appends it to the list.

You can remove batches with the **Remove Batch** button and change the order of batches with the arrow-buttons to the right. To modify the settings of a batch, first select it from the table by clicking on the corresponding row (the row will become bold). As soon as you select a batch, the **Batch Properties** panel at the bottom will show the configuration of it. You may then change the number of runs or the configurations for the phenotype search space, the fitness function, or the initial population. Changes are applied directly to the batch.

Note:
The first batch is created automatically every time you enter this panel or hit the **Finish** button and the list of batches is empty. If you step back to previous panels and change configurations there, all batches that have already been created will remain unchanged. In case you want the batches to run with the new configurations, you have to create new batches instead.

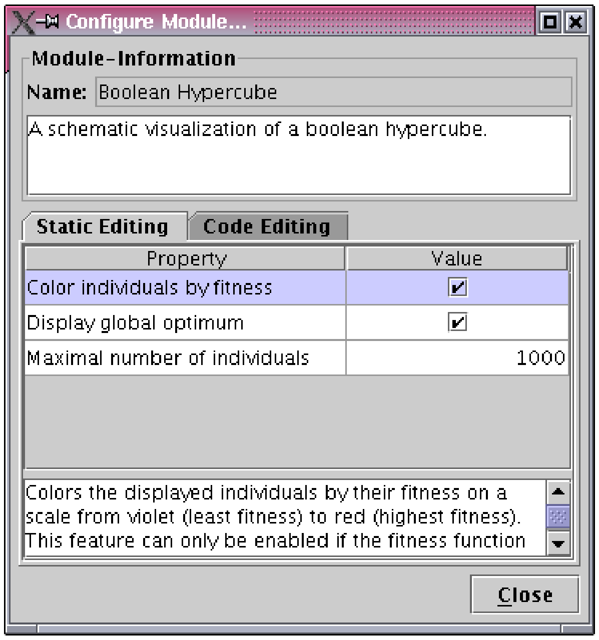
Try adding a few batches with different configurations and press **Finish**.

Well done. Your schedule is now complete. Now you might want to simulate (run) your schedule to see what it really does. This is covered in Chapter 3: Controlling Runs.

2.6 Configuration of Modules

2.6.1 Configuration of Properties

Whenever a module offers the option to be configured, the **Property Dialog** shown in Figure 2.13 (with varying contents of course) pops up:



Name: Boolean Hypercube

A schematic visualization of a boolean hypercube.

Property	Value
Color individuals by fitness	✓
Display global optimum	✓
Maximal number of individuals	1000

Colors the displayed individuals by their fitness on a scale from violet (least fitness) to red (highest fitness). This feature can only be enabled if the fitness function

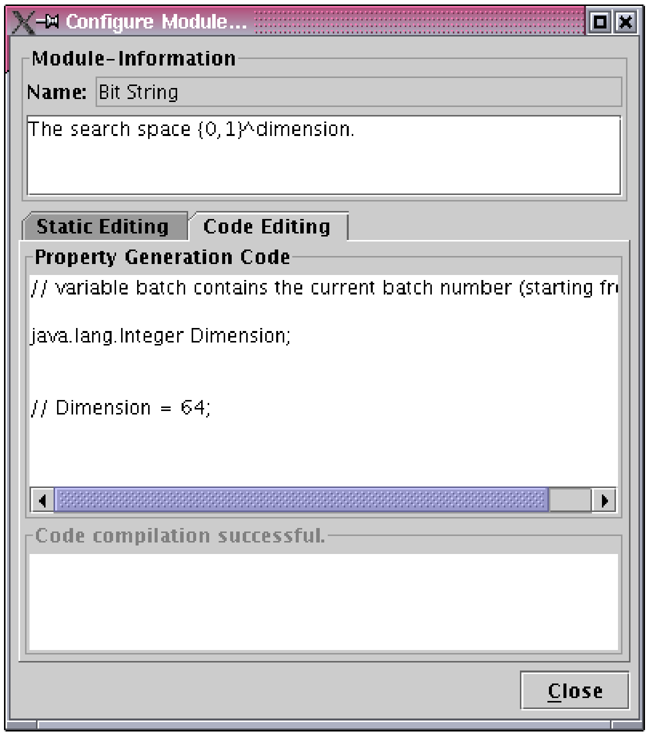
Figure 2.13: Property dialog of view Boolean hypercube.

It displays some information about the module you are configuring, in particular the name and the description of the module as well as the properties the module allows to be configured.

Figure 2.13 shows the property dialog for a Boolean Hypercube. As you might have guessed, you can change the values displayed in the dialog to your needs. Numeric values can be edited, checkboxes can be set or unset, and so on.

2.6.2 Advanced Feature: Code Editing

When choosing the tab **Code Editing**, you will see a large text area where you can use Java code to adjust properties of the configured module. This is an advanced feature that is used to generate properties automatically for multiple batches as the entered code will be applied to every new batch.



The search space {0,1}^dimension.

```
// variable batch contains the current batch number (starting fr

java.lang.Integer Dimension;


// Dimension = 64;
```

Code compilation successful.

Figure 2.14: Code Editing of search space Bit String.

At the top of the panel, you see a Java comment telling you that the variable *batch* contains the current batch number. Below, you see definitions for the variables representing the properties within the module. Note that the variable declarations may differ from their names displayed in **Static Editing**.

Note:
When entering the **Code Editing** panel, all lines of code that set property values are commented out. Be sure to remove the comment signs if you want to write your own property generation code.

At the bottom of the panel, you see another text field informing you about compilation errors. If you finish editing code and hit the **Close** button, FrEAK tries to checks whether the generated code matches the corresponding property types. If you assigned incompatible property types, e. g., by assigning a String value to an integer, FrEAK might displays an error message to inform you of this problem.

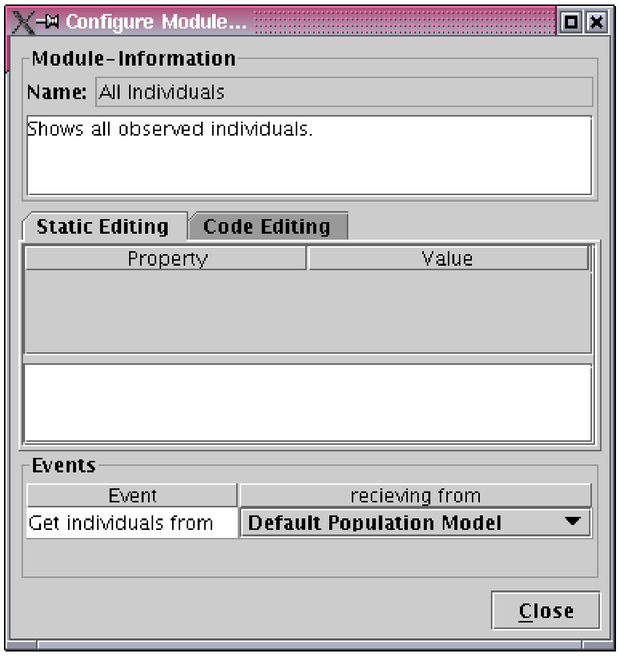
An example for using Code Editing follows. To generate batches with the search space Bit String where the dimension grows in powers of two, you can enter the following code snippet:

```
// variable batch contains the current batch number (starting from 0)

java.lang.Integer Dimension;

Dimension = Math.pow(2, batch);
```

The variable `batch` is used to refer to the index of the batch. If you then close the dialog and create new batches, the i-th batch in the list of batches (starting with 0) will now have a search space dimension of 2^i.

2.6.3 Configuration of Event Sources

Some modules offer the possibility to be registered as event listener for certain events. An example for such a module is an observer that computes data from a specified set of individuals. This observer has to be notified when new individuals are available by an event. In fact, most observers work that way. So where does this event come from? Which other module is the *event source* for this event?


Configure Module...

Module-Information

Name: All Individuals

Shows all observed individuals.

Events

Event	recieving from
Get individuals from	Default Population Model

Figure 2.15: Property dialog of observer All Individuals.

Modules that can be registered as event listeners have a **Property Dialog** as shown in Figure 2.15. (Here, the **Property Dialog** of the All Individuals observer is shown.) You will notice a table of events each displayed with a name and an event source where the module receives this event from. The event source may or may not yet have been set to a valid source (in fact most, if not all, modules will be assigned to meaningful default event sources), but you can change the event source at any time by selecting an item from the drop down box of the event.

Some events allow only one event source (mainly the Schedule itself), where other events even allow *outports* of operators of the graph to be selected as event source. See Figure 2.16 for an example.

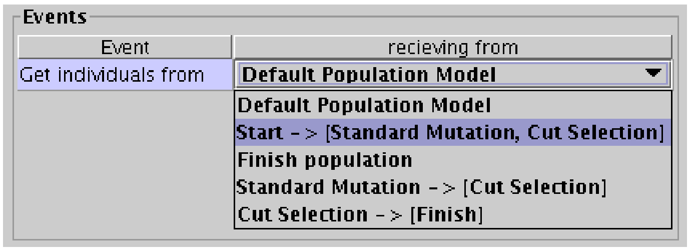


Figure 2.16: Possible event sources for the observer ALL INDIVIDUALS.

For more information about operators and outports, see Section 4.1: Operator Graphs: Concepts.

Of course there are modules that have both properties and event sources to be configured. An example for these modules are the **Parameter Controllers** described in Section 4.3: Parameter Controllers.

If you are done with configuring the module, click on **OK** to return to the **Schedule Editor**.

Chapter 3

Controlling Runs of Evolutionary Algorithms

3.1 Getting Started

When starting FrEAK, you see a window as shown in Figure 3.1.

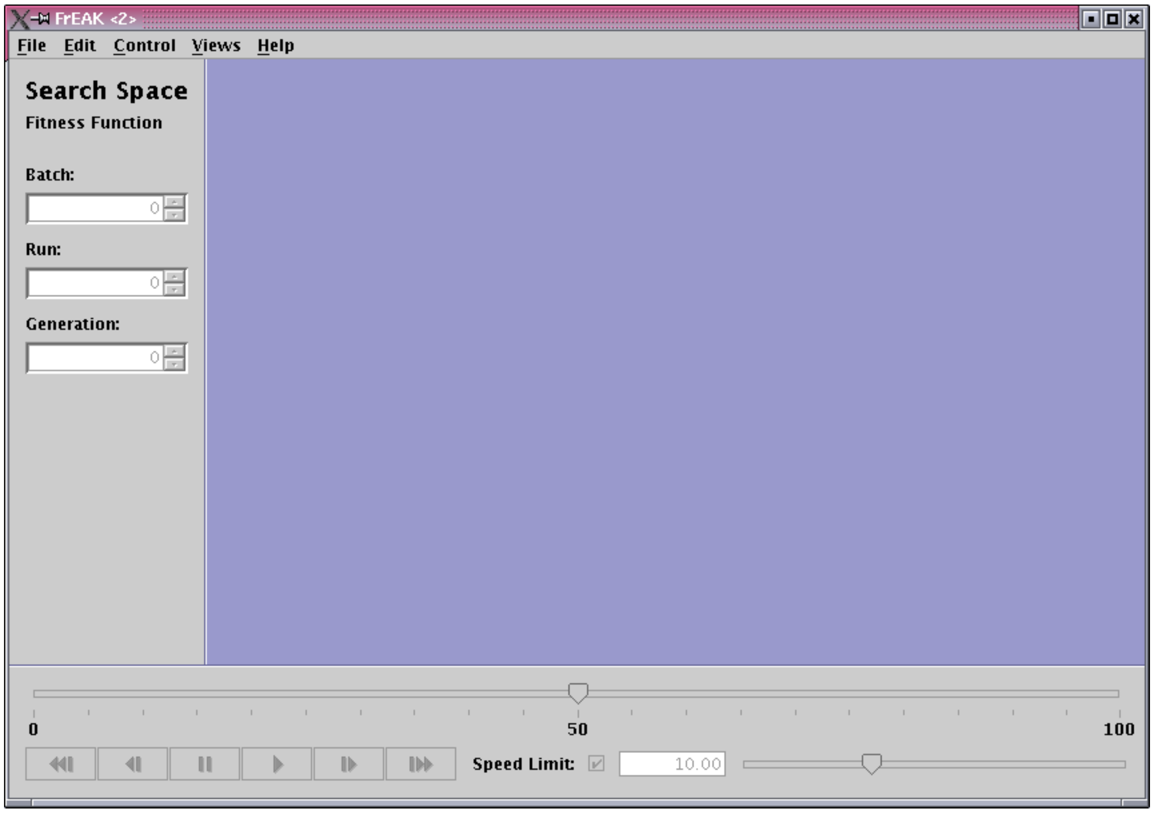


Figure 3.1: The window after starting FrEAK.

You see an information panel on the left side, a control panel at the bottom, and a desktop with the selected views on the right side. The information panel to the left displays the current phenotype search space and the current fitness function. Below, you see the current time index consisting of the current batch, the current run and the current generation.

Note that almost all menu items are disabled at the beginning. The first step to take is to create a new schedule by choosing **New** from the **File** menu. The creation of new schedules is described in Chapter 2: Creating a Schedule. Alternatively, you may open an existing schedule by choosing **Open** from the **File** menu.

The **Help** menu provides further information. You can open this document there or take a look at the **About** dialog.

Figure 3.2 shows the FrEAK window after creating or loading a schedule.

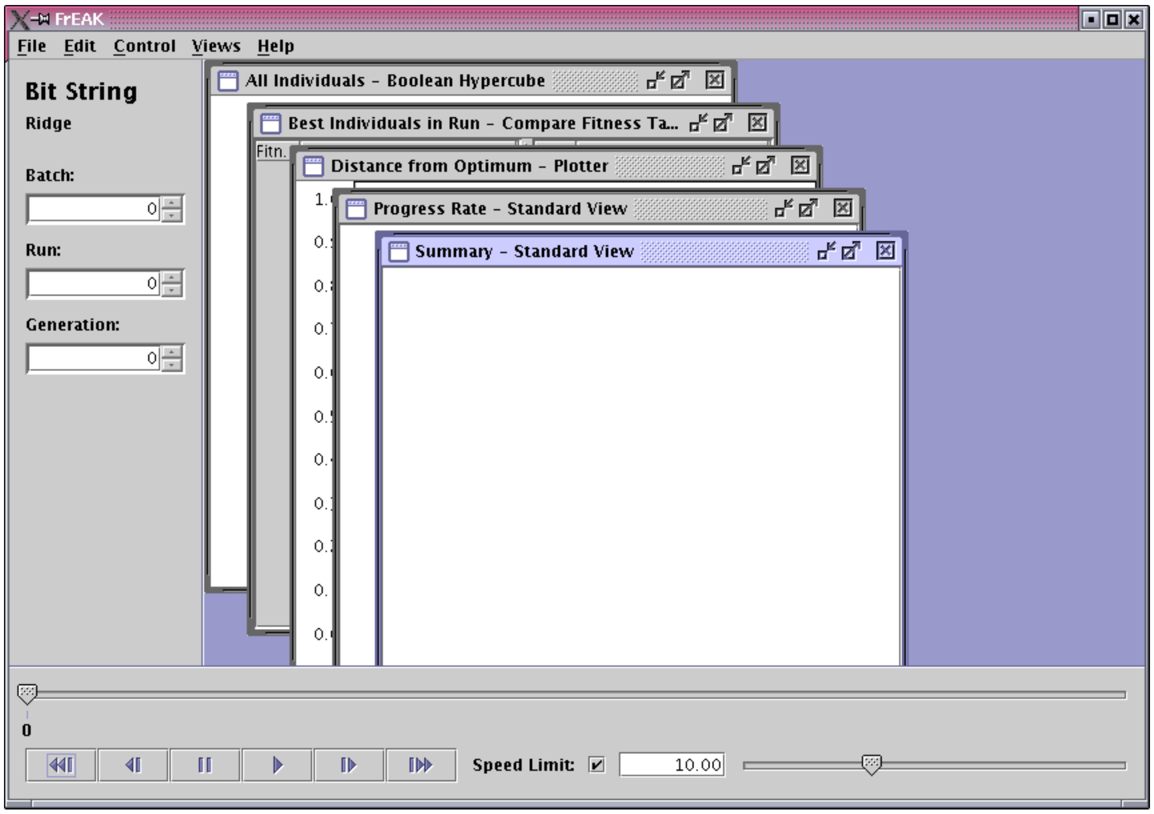

FrEAK
File Edit Control Views Help
Bit String
Ridge
Batch:
0
Run:
0
Generation:
0
All Individuals – Boolean Hypercube
Best Individuals in Run – Compare Fitness Ta...
Fitn.
Distance from Optimum – Plotter
Progress Rate – Standard View
Summary – Standard View
0
Speed Limit:
10.00

Figure 3.2: The FrEAK window after creating a schedule.

In the control panel, you find several buttons used to navigate through the already simulated batches, runs, and generations. These buttons will be explained in the following section. The big slider at the top of the control panel indicates the current generation measured according to all generations that have already been simulated in this run.

When running the simulation you can choose if you want to set a speed limit for the simulation. The speed limit represents the maximal number of generations per seconds and can be enabled and disabled using the corresponding check box in the control panel. The limit itself can be adjusted by using either the text field or the slider to the right.

With a click on the ▶ **Start** button you start the first run.

The simulation can always paused by using the ⏸ **Suspend** button. Some operations are only available while the run is suspended, such as editing the current schedule, modifying the speed limit, current run and current generation, or configuring views.

Press the ▶ **Start** button again and the simulation is continued.

3.2 Watching Replays

FrEAK provides a replay mode to watch past batches, runs, and generations. To step back to an earlier time index, use the navigation buttons in the control panel or enter the desired time index in the information panel on the left. FrEAK automatically switches to replay mode and looks for the specified time index. This may eventually take some time because some runs and generations have to be simulated in the background to reach the desired time index.

The **Control** menu as well as the navigation bar inside the control panel contain the following navigation commands.

- ⏮ **Go to Start** go back to the start of the simulation.
- ◀ **Step Back** go back to the last start of a run.
- ⏸ **Suspend** pause the current Schedule.
- ▶ **Start** start or continue the current Schedule.
- ▶ **Skip Run** jump to the start of the next run.
- ⏭ **Go to End** jump forward to the last known generation.

You can also use the generation slider to quickly jump between generations.

When replay is finished and the last simulated generation is reached, FrEAK automatically turns back to running mode and continues simulating new generations.

Note:
FrEAK uses a deterministic pseudo-random number generator to create sequels of pseudo-random numbers. During replay creation and whenever the schedule is saved to disk, the current state of this pseudo-random number generator is recorded. So, whenever you watch a replay of a simulation, FrEAK uses exactly the same pseudo-random numbers as before. Thus, all the simulation can be reproduced.

3.3 Editing the Current Schedule

You can modify the setup of the current schedule to a certain degree depending on the progress of it. This is done via the same dialog that was used to create a new schedule. To open this dialog, select **Edit Schedule. . .** from the **Control** menu. The handling of this dialog is explained in detail in Chapter 2: Creating a Schedule with the difference that several options are not available and you start with Step 8: Creating Batches of Runs if your schedule was already started.

Note:
If you edit a schedule that was not yet started, the dialog works *exactly* the same as described in Chapter 2. If it was already started, several components will be disabled.

Please refer to Chapter 2: Creating a Schedule on how to work with the schedule editor.

3.4 Menu Items

The menu provides several functions which are described in detail here. Note that there are shortcuts for the most important functions.

3.4.1 File Menu

This menu is used for schedule management and to exit FrEAK.

New Create a new schedule with the Schedule Editor. For details see Chapter 2: Creating a Schedule.

Open Open an existing schedule.

Save Save the current schedule.

Save as Save the current schedule requesting a file name.

Exit Exit FrEAK.

3.4.2 Edit Menu

Contains several operations from the control panel.

Edit Edit the current schedule. For details see Section 3.3: Editing the Current Schedule.

Configure Selected View If the currently selected view is configurable and the current run is suspended, you can configure the view in the configuration dialog by choosing this menu item. For help on the configuration dialog see Section 2.6: Configuration of Modules.

3.4.3 Control Menu

Contains several operations from the control panel. Most of them were already explained in Section Watching Replays.

Start start or continue the current Schedule.

Suspend pause the current Schedule.

Step Back go back to the last start of a run.

Skip Run jump to the start of the next run.

Go to Start go back to the start of the simulation.

Go to End jump forward to the last known generation.

Go to . . . jump to a specified generation.

3.4.4 Views Menu

This menu is used to manage the current views. If a schedule containing views is active, all views are listed in the upper part of the menu and can be activated by choosing the corresponding menu item. The lower part of the menu contains general window management functions for the views.

Tile Tile the views on the desktop. Minimized and closed views are not considered.

Cascade Reorder the views on the desktop in form of a cascade. Minimized and closed views are not considered.

Minimize All Minimize all visible views.

Restore All Restore all closed and minimized views.

Close All Close all opened views. You can restore all view windows with the menu item **Restore All** or simply click on the menu item corresponding to the view you want to restore.

3.4.5 Help Menu

The **Help** menu offers you a global help.

Help opens a HTML version of this document.

About opens the about dialog where you can learn more about the current version and the developer team of FrEAK.

Chapter 4

Designing Algorithms with FrEAK

As mentioned in Chapter 1.2: Evolutionary Algorithms, FrEAK visualizes the control flow of evolutionary algorithms as operator graphs.

This chapter explains the main concepts of FrEAK's operator graphs (Section 4.1: Operator Graphs: Concepts) and describes how to work with the graphical editor for these operator graphs (Sections 4.2: The Graph Editor and 4.3: Parameter Controllers).

4.1 Operator Graphs: Concepts

This section explains the main concepts of FrEAK's operator graphs. If you are impatient, you can skip this section and return to it if you are confused by anything in the following sections of this chapter.

An operator graph visualizes how a given generation becomes the next generation. Operator graphs consist of *nodes* and *edges*. Each node defines an operator that works on some of the individuals of the population. Nodes have *ports* where edges can connect to (see Figure 4.1). Edges define the flow of individuals.

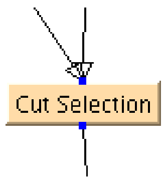


Figure 4.1: An Operators with Ports.

The edges are directed, that means each edge has a source and a target port; those working as source port are called *inports*, those working as target port are called *outports*.

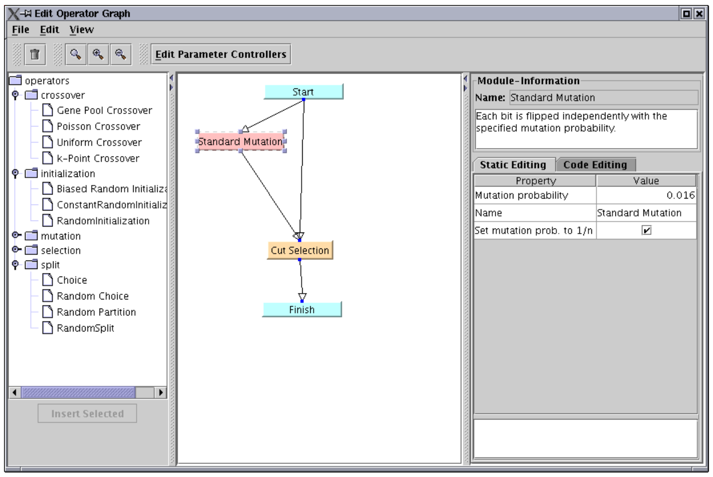
Most kinds of operators have a fixed number of inports and outports. But some kinds of operators, like some split operators, can have any number of incoming or outgoing connections. In FrEAK, these operators are said to have *floating inports* or *floating outports*, respectively.

For instructions how to work with nodes and edges in operator graphs, please refer to Section 4.2.1, Adding and Removing Nodes and Edges.

Another important concept in FrEAK is that of *properties*. Many operators can be configured to some extent, a mutation operator for example may let you specify the mutation probability. Each attribute of an operator that is configurable is called a property. For instructions how to change properties of operators, please refer to Section 4.2.2, Changing Properties of Modules.

4.2 The Graph Editor

The graph editor is divided into three areas (see Figure 4.2).



Name: Standard Mutation

Each bit is flipped independently with the specified mutation probability.

Property	Value
Mutation probability	0.016
Name	Standard Mutation
Set mutation prob. to 1/n	✓

Figure 4.2: The Graph Editor Dialog.

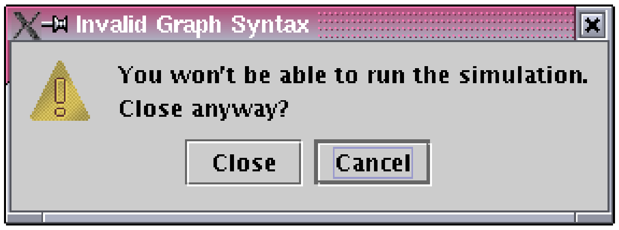
The editor itself is found in the middle. To the left, there is a tree of operators that can be inserted into the graph, and to the right, information about the currently selected module is displayed. You can also edit properties of some of the operators here, see Section 4.2.2.

The menu and the tool bar provide standard editor commands like creating a new, empty operator graph, opening and saving operator graphs as well as zooming the graph display.

Note:
It is only necessary to save the graph if you plan to reuse it later when creating other schedules.

In addition to these commands, you can open the Parameter Controller Setup with the appropriate tool bar button or by selecting **Edit Parameter Controllers** from the **Edit** menu. Section 4.3: Parameter Controllers describes how to work with parameter controllers.

When you have finished creating the operator graph, close the editor dialog by using the file menu's **Close** item. If your operator graph contains syntax errors, you will see a warning message (see Figure 4.3).


Invalid Graph Syntax

You won't be able to run the simulation.
Close anyway?

Figure 4.3: Error Message: Graph has Syntax Errors.

Selecting **Close** and trying to run the resulting schedule will most likely produce errors, so you should select **Cancel** and try to correct the error. Make sure there are no operators that have unconnected, non-floating inports and that the graph contains no cycles.

If the graph doesn't have errors or you have selected **Close** on the warning message dialog, the editor dialog is closed and you return to the schedule editor.

4.2.1 Adding and Removing Nodes and Edges

To add an operator to the graph, simply select it from the operator tree. You can insert the operator by double clicking or by using the **Insert Selected** button.

To connect two operators, select an outport of the source node. Hold and drag the mouse to an inport of the target node and release the mouse to establish the connection.

Note:
The mouse cursor changes to a hand when it is over a port that can be selected in the given situation. Please keep in mind that operator graphs must not contain cycles. You are not allowed to draw edges that would result in a cycle.

It is possible to connect multiple outports to one inport or one outport to multiple inports. If multiple outports are connected to the same inport, all individuals reaching the inport are merged together before being processed by the operator. In case an outport is connected to multiple inports, the outgoing individuals are duplicated for all associated inports.

As explained in Section 4.1: Operator Graphs: Concepts, some operators have a floating number of inports or outports. The graph editor automatically inserts a new, unconnected port whenever a floating port is connected, so you don't have to take care about that. The graph editor also removes the disconnected port when an edge is removed. In other words, operators with floating ports always have an unconnected port.

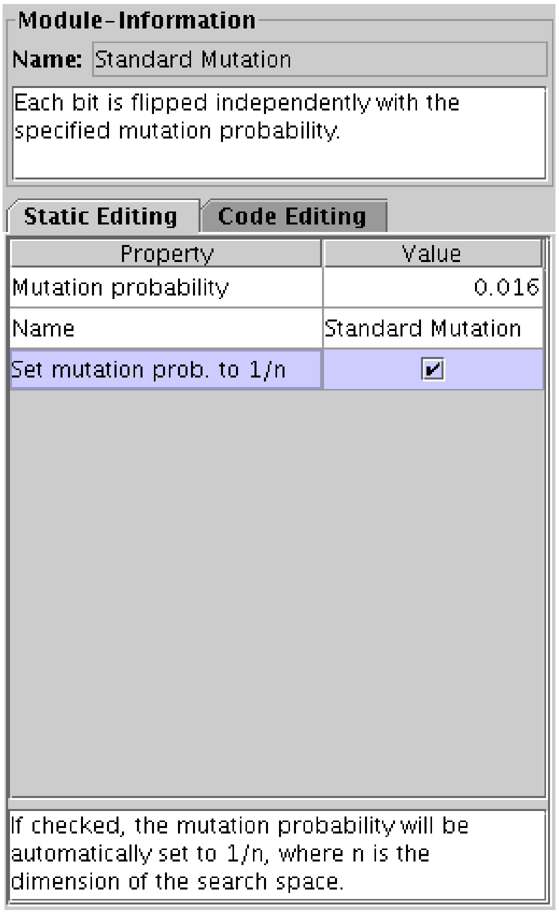
Note:
While "normal" ports are drawn in blue, floating ports are drawn in red so you can recognize them easily.

To remove an element of the graph, select it with the mouse and press DEL or use the appropriate item from the tool bar or the menu. You can also click and drag the mouse to select multiple elements.

Note:
You cannot delete the Start or Finish operators.

4.2.2 Changing Properties of Modules

To the right of the graph editor itself is the property inspector of the operator that is currently selected in the operator graph (see Figure 4.4).


Module-Information

Name: Standard Mutation

Each bit is flipped independently with the specified mutation probability.

Property	Value
Mutation probability	0.016
Name	Standard Mutation
Set mutation prob. to 1/n	✓

If checked, the mutation probability will be automatically set to 1/n, where n is the dimension of the search space.

Figure 4.4: The Property Inspector

This inspector displays some information of the operator, such as its name, its description, and the description of its ports.

Underneath the description of the ports, you can see and edit the current values of the operator's properties. To edit a numeric or textual property, just doubleclick into the appropriate cell in the table and type the new value. Confirm your changes by pressing Enter. Values that can only be either *true* or *false* can be set by checking or unchecking the checkbox.

At the bottom of the property inspector, the description of the selected property is displayed.

Note:
In most cases, if you enter invalid property values, the module takes care of it immediately and reset the property to either the old value or a corrected value. In some cases, modules with a rather complex structure and dependencies among the properties might wait until all changes have been made and then perform a check. If the properties are found to be invalid, an error message might be displayed. In such a case, the current property inspector remains active and waits until the invalid properties have been corrected, even if you select another module meanwhile.

4.3 Using Parameter Controllers

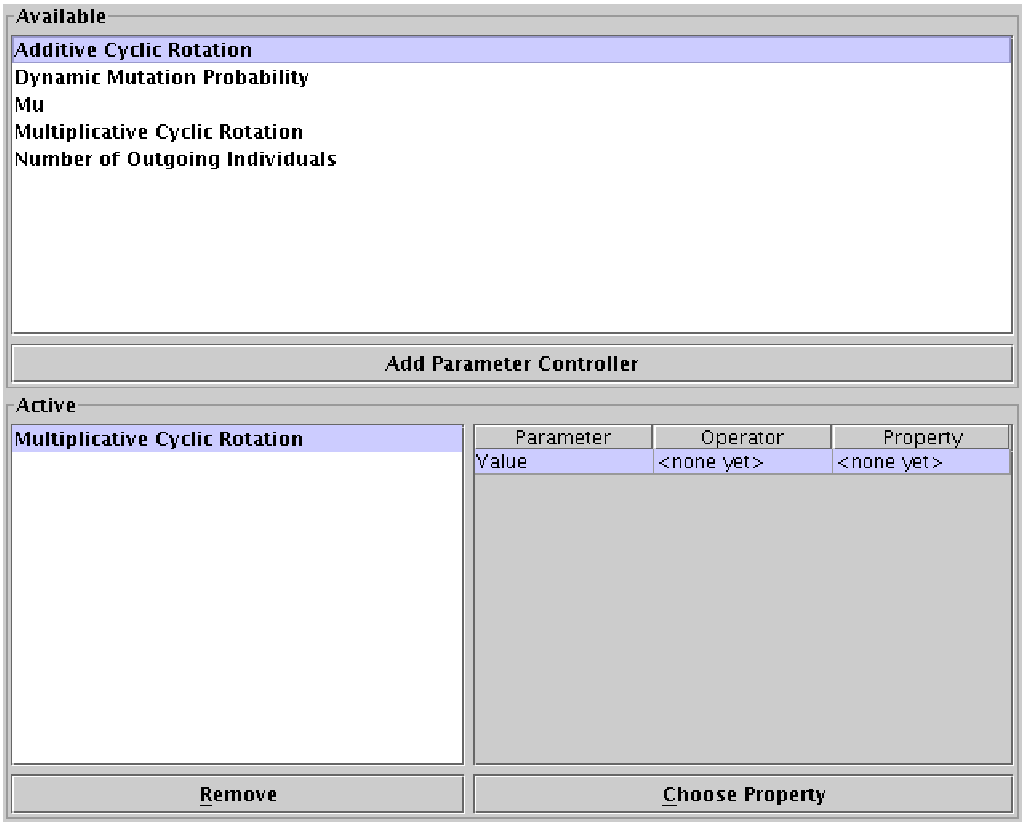
The concept of parameter controllers has been added to FrEAK to account for adaption. A parameter controller contains some attributes which in this context are called *parameters*. A parameter controller can listen for some events, and during the handling of an event it can modify its parameters.

What is important is that the changes in the parameters are reflected in properties of operators in the graph. To be more precise, an operator and one of its properties can be assigned to a parameter. When the value of a parameter changes, the associated property is set to the same value.

4.3.1 Adding Parameter Controllers

You can open the parameter controller setup dialog from the graph editor dialog with the appropriate tool bar button or by selecting **Edit Parameter Controllers** from the **Edit** menu. In the left part of this dialog, you can see the interface for adding and removing parameter controllers (see Figure 4.5).

You can add parameter controllers from the available list to the active list (by double clicking on them or by selecting and clicking the button between the two lists), and also select parameter controllers from the latter list and remove them by clicking the button labeled **Remove**.


Available
Additive Cyclic Rotation
Dynamic Mutation Probability
Mu
Multiplicative Cyclic Rotation
Number of Outgoing Individuals
Add Parameter Controller
Active
Multiplicative Cyclic Rotation
Remove
Choose Property

Parameter	Operator	Property
Value	<none yet>	<none yet>

Figure 4.5: Adding parameter controllers.

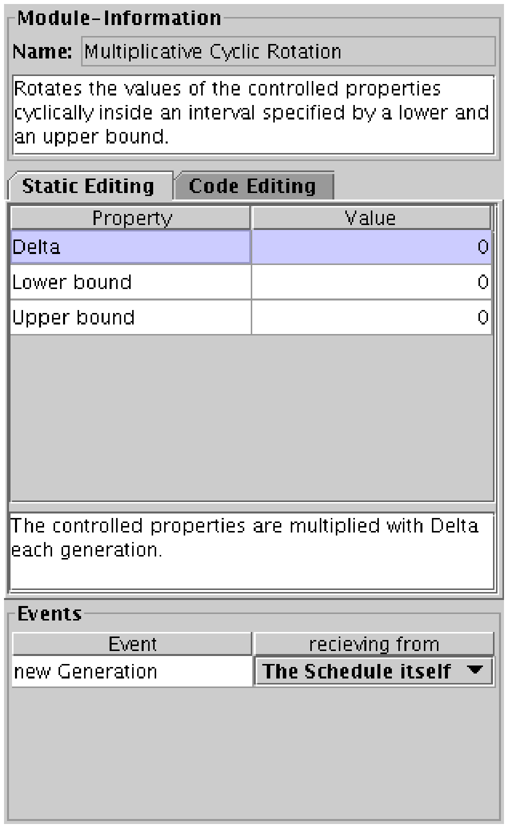
Once you have added a parameter controller, you can configure it. This means mainly setting the sources for the events it can listen to and assigning operators with their properties to the parameters. Furthermore, the functionality of the parameter controller may be parametrized; in this case, you can modify some properties.

4.3.2 Assigning Event Sources and Configuring Parameter Controllers

If you click on a parameter controller in the active list, its properties and events will be displayed in the module information panel at the right of the parameter controller setup dialog (See Figure 4.6). In the subpanel labeled **Events** you find a table with one line for each event the parameter controller can handle, containing the description or name of the handled event and a drop down box which can be used for setting the event source for the event. Section 2.6.3: Configuration of Event Sources describes in detail how to specify event sources.

Above, you find the subpanel labeled **Properties** in which another table containing the properties of the parameter controller is located. Here you can set these properties and thus modify the parameter controller's behaviour.

Note:
The **Properties** subpanel is only available if the selected parameter controller is configurable.



Name: Multiplicative Cyclic Rotation

Rotates the values of the controlled properties cyclically inside an interval specified by a lower and an upper bound.

Property	Value
Delta	0
Lower bound	0
Upper bound	0

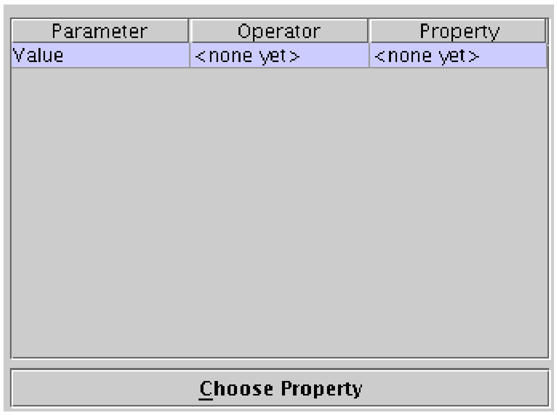
The controlled properties are multiplied with Delta each generation.

Event	recieving from
new Generation	The Schedule itself

Figure 4.6: The information panel for the selected parameter controller.

4.3.3 Assigning properties to parameters

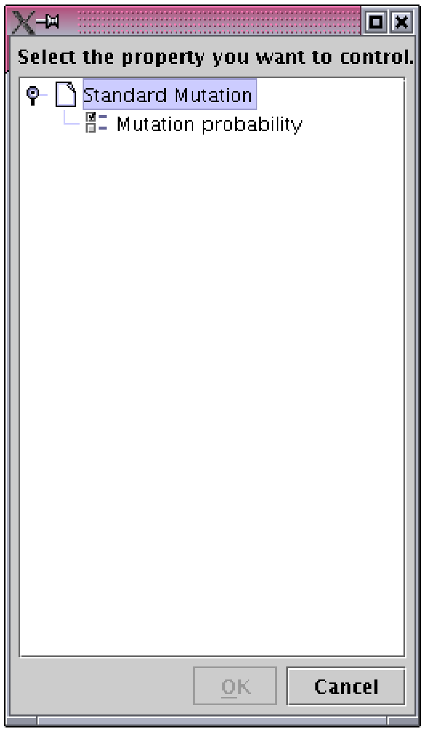
If a parameter controller is selected in the active list, a table containing its parameters is displayed at the right of the parameter controller main panel (see Figure 4.7). To assign a property to a parameter, you must select the parameter in the table and click the button **Choose Property**.



Parameter	Operator	Property
Value	<none yet>	<none yet>

Figure 4.7: Setting up properties.

After doing this, you will see a window like the one in Figure 4.8. There is a list of all operators in the graph which contain properties compatible with the parameter you have selected. You can open the list of properties for each operator by clicking on the symbol left to an operator's name. After this, select the property you want and click **OK** to make the selection take effect.



Select the property you want to control.

Figure 4.8: Selecting properties to control.

Index