The ASSIST-system

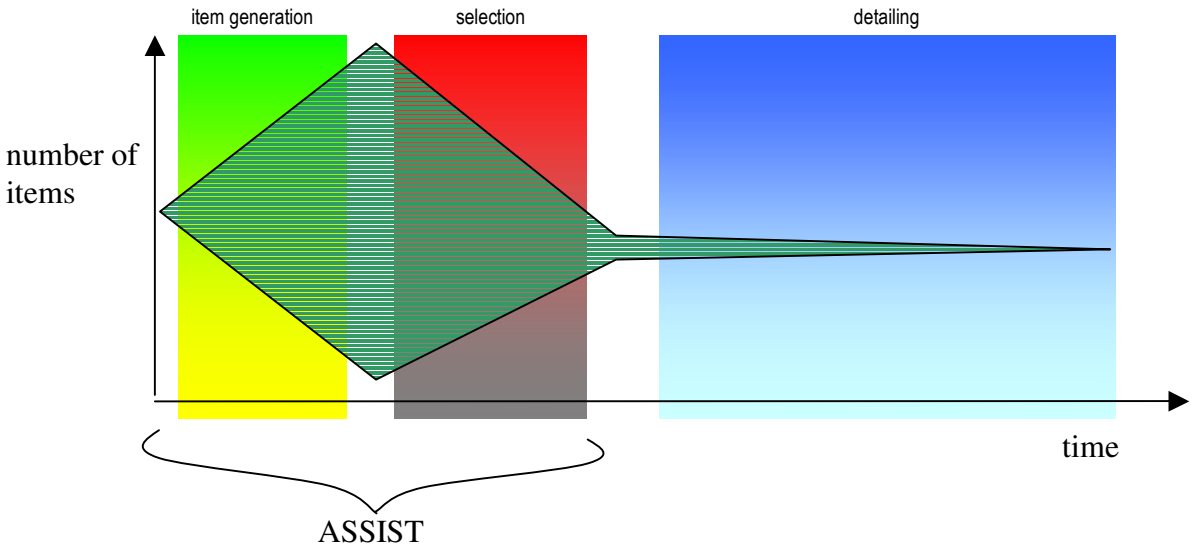
General Introduction to ASSIST

Design Processes

A simple model of a design process consists of 3 phases:

- item generation (or: *idea generation* or *concept generation*)
- selection
- detailing

Schematically, these phases can be depicted as



In the item generation phase, it is essential to create as many as possible ideas (hence the colour green: this relates to creativity), and criticism is absolutely forbidden (hence the colour yellow: this relates to optimism).

In the selection phase, initial ideas are critically assessed (hence the colour black: this relates to pessimism or constructive negativism), where criteria should formulated to make intuition more explicit (hence the colour red: this relates to intuition). The large collection of ideas from the item generation phase needs to be brought back to the very few very best.

In the detailing phase, the few remaining items are cast in the form of (mathematical) models (hence the colour blue: this relates to systematic thinking) and factual information is brought in to help making realistic estimates for crucial dimensions in the item-to-be-realized (hence the colour white: this relates to objective facts).

The tool-stack ASSIST contains a number of instruments to facilitate the first two phases; for support in the detailing phase, the tool microAccel (see elsewhere) may be used.

The instruments in ASSIST

ASSIST is an acronym for ***A***SSIST ***S***upports ***S***ystematic ***I***nnovation by means of ***S***oftware ***T***ooling. The instruments currently available in ASSIST are the following:

- File I/O: create a new project which is initially empty, or load an existing project from an earlier session, or write the current project to background storage for future processing
- Build Inspirators: bring in domain-specific knowledge to drive a grammar-based system to proposes inspiring suggestions
- Inspire: create new items, either from scratch, either as a result of an inspiring suggestion, or either as a result of systematic, new combination of values for classifiers
- Build Classifiers: develop an ontology where the items are described in terms of distinguishing classifiers
- Classify: given the ontology, label each item with the relevant values for each of the classifiers
- Build Ranking System: input attributes that are used to assess the quality of items, and state rules to express that values of certain items are better with respect to some attribute than others
- Rank: perform the ranking of the items in terms of the ranking attributes
- Auto Weights: given the ranking values for a number of items, and given their total quality-scores, automatically calculate (approximate) weights that can be used to express differences in importance for various ranking attributes
- Cluster and visualise: aid the interpretation of the present set of items, their classification and their attribute-ranking by means of intuitive visualisation.

In the sequel, each of the tools is described in detail.

How to get started

ASSIST is implemented as a series of interactive forms, each form representing one of the tools. The forms allow processing of a number of tables (including a table of items with their classification and their ranking, and a table containing the inspiration grammar). These tables are stored in MS-EXCEL worksheets, one table per sheet. They can be inspected, and some of the information in the tables is visualized by means of MS-EXCEL graphs. The contents of the worksheets should never be edited or manipulated directly, however: the only access of the information in the tables goes via the forms.

The worksheets that need to exist for ASSIST are:

- Items – this contains the table of items, together with the classification and the ranking
- Inspire – contains the grammatical structures (so called terminals and non-terminals) that are used to form the inspiring suggestions
- Results – for displaying the numerical values in terms of EXCEL-graphs

A menu, named *assist* gives access to the forms. At any time, at most one form is open. If a tool is selected from the *assist*-menu, the associated form is presented, and the relevant worksheet is selected and made visible (either "Items" or "Inspire" – the "Results"-sheet is only kept for internal processing). In order to inspect another worksheet, the form first should be closed. Since the entire project is realised as an MS-EXCEL workbook, it can be saved, opened, and renamed just as any other MS-EXCEL document. Projects and sets of inspiration-grammar-rules are stored as text files, with the extensions '.amf' (for Assist Model File), and '.igf' (for Inspiration Grammar File). Although all processing takes place via the various forms of ASSIST, these files can be opened by any regular text editor. Modification of these files *outside* ASSIST may jeopardize their integrity, though, which can cause ASSIST to fail.

The *assist* menu is added to the standard MS-EXCEL menus (such as *file, edit, view, ...*). Upon opening an ASSIST-application it appears automatically on the EXCEL menu bar; further, the ASSIST file I/O form is opened, since an ASSIST session typically starts with reading in a previous project or defining a new project from scratch.

FILE

General introduction

A new project starts by activating the initialise-function. This clears all the required worksheets. In order to prevent valuable information from a previous run to get lost, the user is asked to confirm clearing the worksheets. As long as one or more of the worksheets of a previous session is not cleared, however, ASSIST may not function properly. For a consistent session, it is important that all worksheets are cleared.

Manual

The form looks as follows:

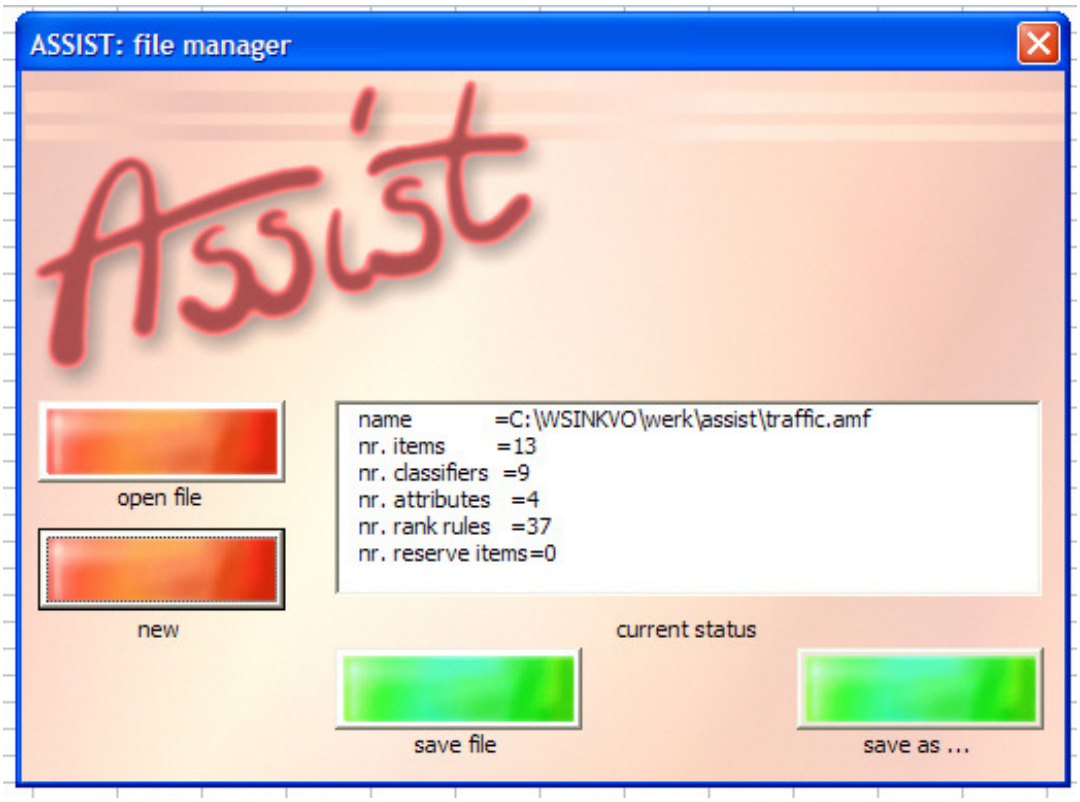


Figure 1

screenshot of the File I/O-form

Controls

Open file

This opens a standard MS-Windows file browser which allows to navigate the file system and to select an existing '.amf'-file to be read in. In case an unsaved project exists, the user is asked first if this project should be saved.

Current status

This text field gives the information of the currently opened project: its title (=the name of the file on disk, including the full path), the number of items, the number of classifiers, the number of attributes, the number of ranking rules (see below), and the number of reserve items (see below).

New

In order to start a new project, the New-button should be clicked. If an unsaved project exists, the user is asked if this should be saved first.

Save file

In order to save the current project under its current name, the Save file button should be clicked. This replaces the existing file with the current name by the current contents of the project.

Save as ...

In order to save the current project under a new name, the Save as... button should be clicked. This pops up a standard MS-Windows file browser which allows the user to navigate the file system and to select an existing filename or to type in a new name. If an existing file is given, ASSIST asks for confirmation first.

Build Inspirator

General introduction

Items can be entered into the ASSIST-system in one of three possible ways: either as idea-from-scratch; either as inspired by a suggestion that is proposed by ASSIST, or as a new combination of classifier values (to be explained later). The 'inspiring suggestions' that are produced by ASSIST result from a grammar with so-called terminals and non-terminals. Terminals are words or expressions in a particular category, that can be substituted for each other; this category proper is identified by a non-terminal. For example: in order to make expressions such as 'dark green', 'light red', 'halfway between orange and brown', et cetera, we can make a grammar starting from non-terminal `<colourExpression>`. To produce actual colour expressions, we can substitute various expressions for this non-terminal. These expressions are called *terminals*[1] Examples are:

```
(1) purple
(2) <basicColour>
(3) <modifier> <basicColour>
(4) halfway between (<colourExpression>) and (<colourExpression>)
```

In this case, the non-terminal `<colourExpression>` comes with a list of 4 terminals, and three of them contain further non-terminals. Example (1) allows `<colourExpression>` simply to be replaced by the word `purple`. In example (2), the `<colourExpression>` is replaced by another non-terminal, namely `<basicColour>.` Suppose that for the non-terminal `<basicColour>` we would have the terminals

```
red
green
yellow
white
```

then by means of the substitutions `<colourExpression>` → `<basicColour>` → `red,` the final result could be "red". The last step of these substitutions is called a *resolution*: a non-terminal is substituted by an expression which contains no further non-terminals. If the non-terminal `<modifier>` would be given a list of expressions like

```
dark
bright
transparent,
```

then example (3) could produce "bright yellow" via the substitutions `<colourExpression>` → `<modifier><basicColour>` → `bright yellow.` Finally, example (4) is a very powerful production rule. A possible chain of substitutions, for instance, could be `<colourExpression>` → `halfway between (<colourExpression>) and (<colourExpression>)` → `halfway between (<basicColour>) and (halfway between (<colourExpression>) and (<colourExpression>))` → `halfway`

[1] In the theory of grammars, the word 'terminal' is reserved for expressions that contain no non-terminals. We are a little bit sloppy in this respect: we say that every production rule is of the form <non-terminal> → <terminal>, where <terminal> may contain further non-terminals.

between (red) and (halfway between (<modifier><basicColour>) and (purple)) → halfway between (red) and (halfway between (dark green) and (purple)).

We see that by providing suitable terminals and non-terminals, arbitrary complicated constructions can be obtained. Every time when the user clicks the button ask inspiring question in the Inspire-form (see below), a chain of substitutions is performed, starting from the non-terminal <inspiration>. Which substitutions occur is determined by chance: a random production rule is selected from the list of available terminals for any non-resolved non-terminal. It is possible that a non-terminal has no list of associated terminals; in that case, the non-terminal cannot be resolved, and an expression of the form < ...> will appear in the final result.

In ASSIST, there is one non-terminal 'hard-coded' into the system. This is the non-terminal <inspiration>. All further terminals and non-terminals need to be given by the user, since they will be different from project to project.

Manual

The form looks as follows:

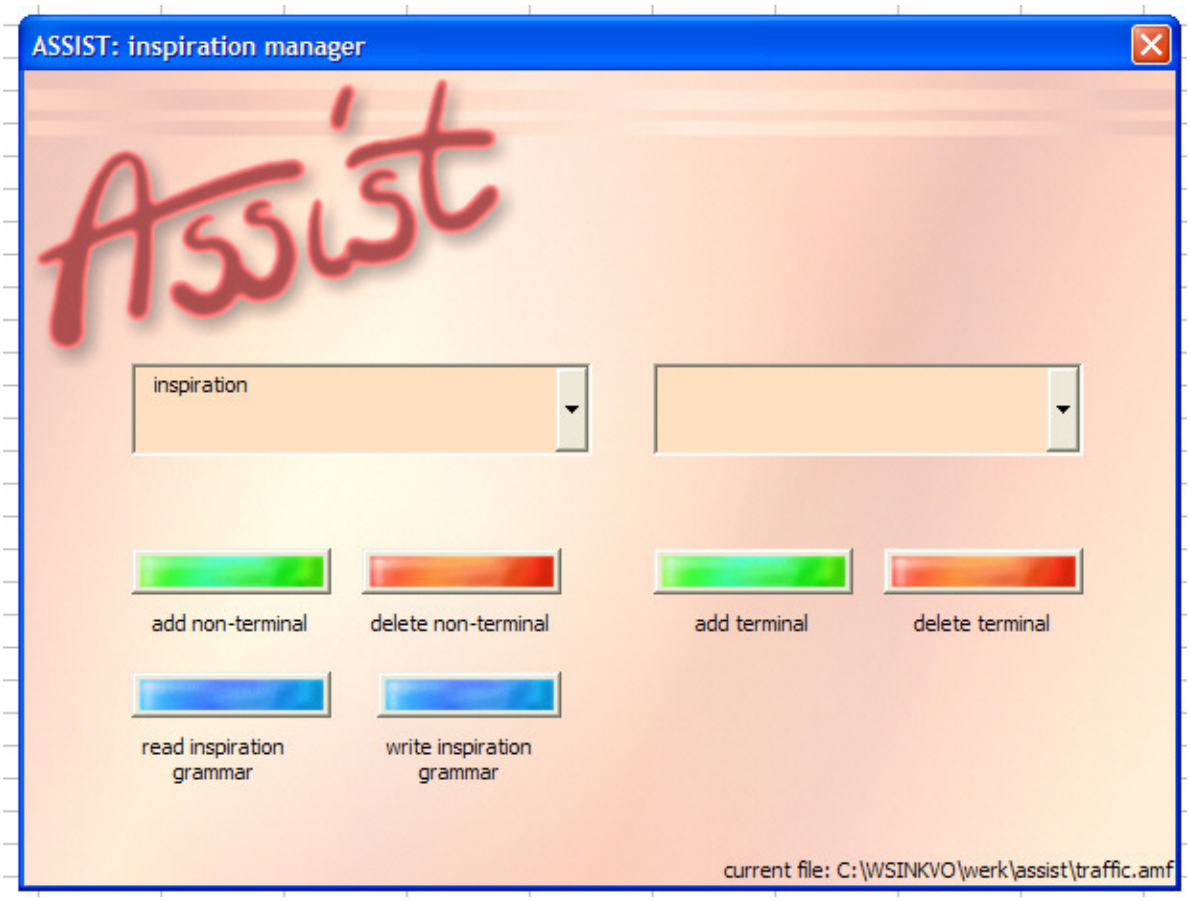


Figure 2

screenshot of the Inspiration Manager-form

Controls

the left selection list contains the present collection of non-terminals. If an item of the list is selected (by means of a pull-down menu, obtained by clicking the down-arrow button right from the list), the associated terminals appear in the right selection list. The selected non-terminal appears in the textbox. (It cannot be edited, though: it can only be deleted.)

the right selection list contains the collection of terminals associated to the presently selected non-terminal in the left selection list. One of them can be selected, in order to be deleted (by clicking the delete terminal button, see below).

add non-terminal : In order to add a non-terminal, the text box of the left selection list needs to be emptied (select the string in the textbox and type 'del', or just type over it), and a new string is entered in the textbox. If this string contains one or more sub strings between < ...>, and these sub-strings do not occur yet as non-terminals, they are interpreted as new non-terminals, and they are automatically created. The name of a new non-terminal doesn't need to be put within < ...>.

delete non-terminal: the presently selected non-terminal is deleted from the list. All terminals that were associated with it are deleted as well. The non-terminal `<inspiration>`, which forms the starting point of the entire grammar, cannot be deleted however.

add terminal: the expression typed into the textbox of the right selection list is added as a terminal for the non-terminal that is presently selected in the left selection list. If this expression contains a sub string between < ...> that does not yet occur as non-terminal, it is automatically created as non-terminal. A terminal is at least two characters long.

delete terminal: this deletes the terminal that is presently selected in the right selection list. By clicking the delete terminal-button, no non-terminals ever will be deleted: even if the last terminal associated to a certain non-terminal is removed, this non-terminal remains in the list.

read inspiration grammar: compiling a meaningful inspiration grammar can be quite an amount of work. Therefore, ASSIST allows inspiration grammars to be saved and restored for later use. Inspiration grammar files are text files with extension ".igf". Clicking the read inspiration grammar-button opens a file dialog to search and select .igf-files. The contents of the opened .igf-file is merged with he current of the grammar: non-terminals and terminals are only added if they do not occur yet.

write inspiration grammar: the current inspiration grammar is written to a file. The user is asked for a filename (default name is "inspiration", or the latest filename chosen for writing). If this filename exists, a warning is issued.

Inspire

General introduction

The list of items is the central ingredient of the ASSIST system. The Inspire-tool serves to manage this list. It allows to add and delete items. An item is a string, the *name* of the item. Further, an item can have an optional *comment* string which may contain additional information. Comments are not used in the ontology or the ranking in the ASSIST-system; they only serve for clarification to the user. Items can be deleted for good, or they can be moved to the reserve list. Items on the reserve list are not taken into account in the ontology, and they are not ranked, but they can be restored at a later stage. For easier reference, items are put in the table on the items-worksheet where they can be overlooked at a glance.

Manual

The form looks as follows:

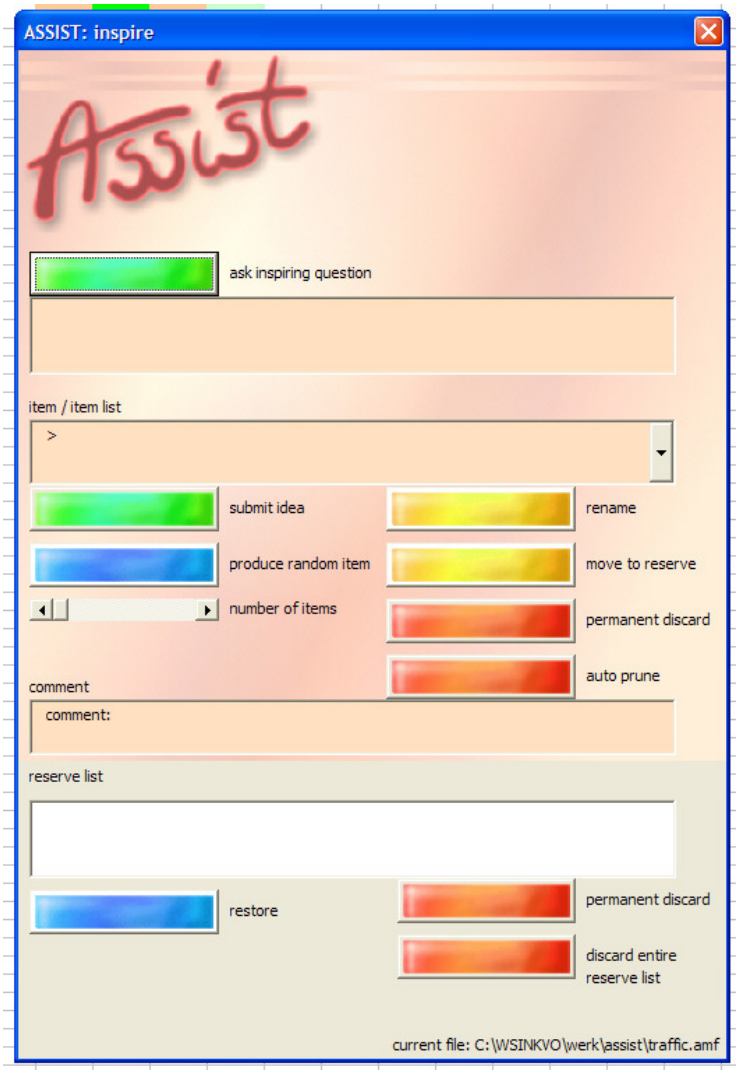


Figure 3

screenshot of the Inspire-form

The form consists of roughly three parts. The top part deals with generating inspiring suggestions; the middle part allows to input items, or to generate items by new combinations of classifier-values in the ontology (see below). The lower most part gives control over the reserve list mechanism.

Controls

ask inspiring question: clicking this button produces a string in the textbox for inspiring suggestions which results from a random chain of substitutions of non-terminals by terminals, starting from the non-terminal `<inspiration>`, as explained in the manual for the Build Inspirator-form

textbox for inspiring suggestions: contains the most recent (inspiring) suggestion. If the produced string is too long to be entirely visible, you can click in the textbox and navigate with the right- and left arrow buttons on the keyboard to read all of it.

item/item list: a combination of textbox and pull-down list. Added items are appended to the tail of the list. Operations such as rename, discard or move to the reserve list apply to the selected item in the list. The entire contents of the list can be visualised by clicking on the down-arrow button right from the text box.

submit idea: an idea is a string typed in the textbox of the item/item list. Clicking the submit idea-button enters this string into the item-list. If also some text is typed in the comment textbox, this text is stored in the comment-field of the new item.

rename: if there is a selected item in the item/item list text box, clicking on the rename button makes a dialog box appear that allows to type a new name for the item. This is particularly useful for the automatically generated items, since these have non-descript names such as <***> followed by a four-digit random number. Also, a new entry for the comment-field can optionally be given.

produce random item: if an ontology has been built (that is, if classifiers and classifier values have been defined, see below), ASSIST can try to form new combinations. If this succeeds (within 10000 tries), a new item is created with the name <***><number> where <number> is a unique four-digit random number. This new item has no comment: ASSIST of course has no idea what this combination of classifier values would amount to. It only serves as a hint for the user; the user should seek to interpret this (novel) combination of classifier values in terms of some meaningful idea or concept. It is possible, of course, that no new items can be generated. For instance, if the ontology consists of 2 classifiers, each with 3 values, no more than 3 x 3 = 9 distinct items exist. If all these combinations already occur in the ontology, no new individuals can be created. In that case, a message pops up "no new item could be generated". An other situation could theoretically occur if there are very many classifiers, possibly with many different values, and a *very* long list of items in the ontology, that new combinations theoretically exist, but the applied (randomised) heuristic doesn't succeed to find any new one. It is questionable, though, if a (human) user would be able to make sense of this extremely large pool of ideas ...

slider number of items: clicking the button produce random item by default produces one new item; with this slider, however, we can ask for several (>1) new items at once. All these items,

if possible, will be subjected to automatic ranking (see the mechanism for auto-ranking, described below)

move to reserve: clicking this button moves the selected item to the reserve list. This is typically done for items that correspond to ideas that produce a very low ranking, or that fall below the Pareto front. Items that were automatically generated by ASSIST, and that therefore have no further intentional interpretation (other than a set of values for the classifiers in the ontology) usually will not be moved to the reserve list: they will be simply be permanently discarded if they don't have a sufficiently high ranking (see the mechanism for auto-ranking, described below).

permanent discard (1): clicking the permanent discard button above the comment-textbox discards the presently selected item in the item/item list without moving it to the reserve list.

auto prune: this moves all items that are not on the Pareto front to the reserve list. Notice: ASSIST does not check if a new item name is different from all the item names in the reserve list. It is therefore possible that two items in the reserve list have the same name, which means that they cannot be distinguished by ASSIST. The user should make sure that names are meaningful and different!

comment: this textbox can be used to enter a comment string for the presently added item. Clicking submit idea stores the text in the comment-box together with the new item in the item list.

reserve list: this list contains the items that had been added in the past, but that are currently not part of the ontology, nor take part in the ranking. The buttons restore or permanent discard operate on the selected item in this list. The entire list is shown by clicking on the down-arrow button right from the text box.

restore: this button moves the presently selected item in the reserve list back to the item list.

permanent discard (2): the presently selected item from the reserve list is permanently removed if the permanent discard-button below the reserve list is clicked.

discard entire reserve list: this discards the entire contents of the reserve list.

Build Classifier

General introduction

We study the following design problem: a manufacturer for hardware tools produces a large variety of equipment: hammers, saws, drills, pliers, … To increase the market, they consider to expand their product range. A traditional way to do so is to set up a brainstorm – however, the results from a brainstorm are just a haphazard collection of samples of an infinitely large space of possibilities. It would be much more efficient if we could provide this space with a map, or a system of coordinates that would allow systematic navigation. That would help us to find empty regions, i.e., options for innovation or expansion.

Such a coordinate system, or *ontology*, can be found if we endow the space of possibilities with *orthogonal, operational classifiers*, where for each classifier the set of values can be finitely *enumerated*. Here, an *classifier* is a function that maps an item to the value that is assumed by the classifier when applied to the item. For instance, the classifier 'colour' maps a cucumber to the value 'green', and a banana to the value 'yellow'. Two classifiers are said to be *orthogonal* if all combinations of their values have more or less equal probability to occur; in other words, if the value of one classifier does not predict the value of the other classifier. For instance, 'contains sugar', with values 'much', 'little', and 'no' is a strong predictor for the classifier 'taste', with values 'sweet', 'sour', 'bitter' and 'salt', since food that contains much sugar tastes sweet. Therefore, 'contains sugar' is not very orthogonal to 'taste'.

A classifier is *operational* if it applies to all items in the considered domain. For instance, if 'software' would be one of the tools in the list, the classifier 'weight' would not be operational, since software cannot be mapped to an amount of kilograms (it can be mapped to an amount of kilobytes, though …). If finding the value of a classifier when applying it to an item of the list requires a substantial amount of discussion, this often means that the classifier is not very operational.

Finally, a classifier is said to have a *finitely enumerated set of values* if we can produce a list of all outcomes of that classifier. A trivial case is a *Boolean* classifier that can only assume the values 'true' or 'false' (such as the classifier 'is legal', 'is a banana', 'is driven by a motor', ...). A nice example of a 3-valued classifier with values that can be enumerated is 'aggregation phase', with values 'gaseous', 'liquid', or 'solid'. But classifiers such as 'shape' or 'is used for' are not finitely enumerated.

Finding classifiers is not trivial, and finding classifiers that are reasonably orthogonal is sometimes quite tough. Still, the endeavour is worth the effort. Indeed, if a good set of classifiers can be found for a given domain (in our example, the domain of hardware tools), this set of classifiers can be said to represent (a substantial part of) the *world knowledge* or *domain knowledge* or *ontology* underlying the list of items. Such a set of classifiers, among other things, can be used to see if new combinations of values might yield a novel product.

A 'good' set of classifiers for any given set of items should be sufficient to distinguish all items. Indeed, if a present set of classifiers is insufficient to distinguish any pair of items, whereas we *know* that these two items are different for some reason, we can argue that there is still some chunk of our world knowledge that is not yet captured in he set of classifiers, and it is challenging to try to find the associated missing classifier. At the same time, a 'good' set of classifiers should be as small as possible. Indeed, for every next classifier the requirement of

(near) orthogonality with all earlier classifiers is increasingly more challenging, and the fewer classifiers are needed, the least redundancy has to be coped with. (Two classifiers are said to be *redundant* if they are not very orthogonal).

Finding a good set of classifiers is partially a matter of creativity, and the ability to think in abstract terms about an application domain. It is, however, also an administrative burden: verifying the orthogonality of two (meaningful) classifiers is a matter of verification, bookkeeping and calculation. The latter task, therefore, can and should be automated, thus permitting the (human) user to focus on the first task.

The Classify-tool, together with the Build Classifier-tool form the components in the ASSIST-toolkit that perform the administrative duties of building orthogonal sets of classifiers.

The tool Build Classifier allows to enter and administrate classifiers, and for each classifier, the set of allowed values. In the tool Classify, these classifiers and classifier values are used to build an ontology for the items in the items list.

Manual

The Build Classifier-form looks as follows:

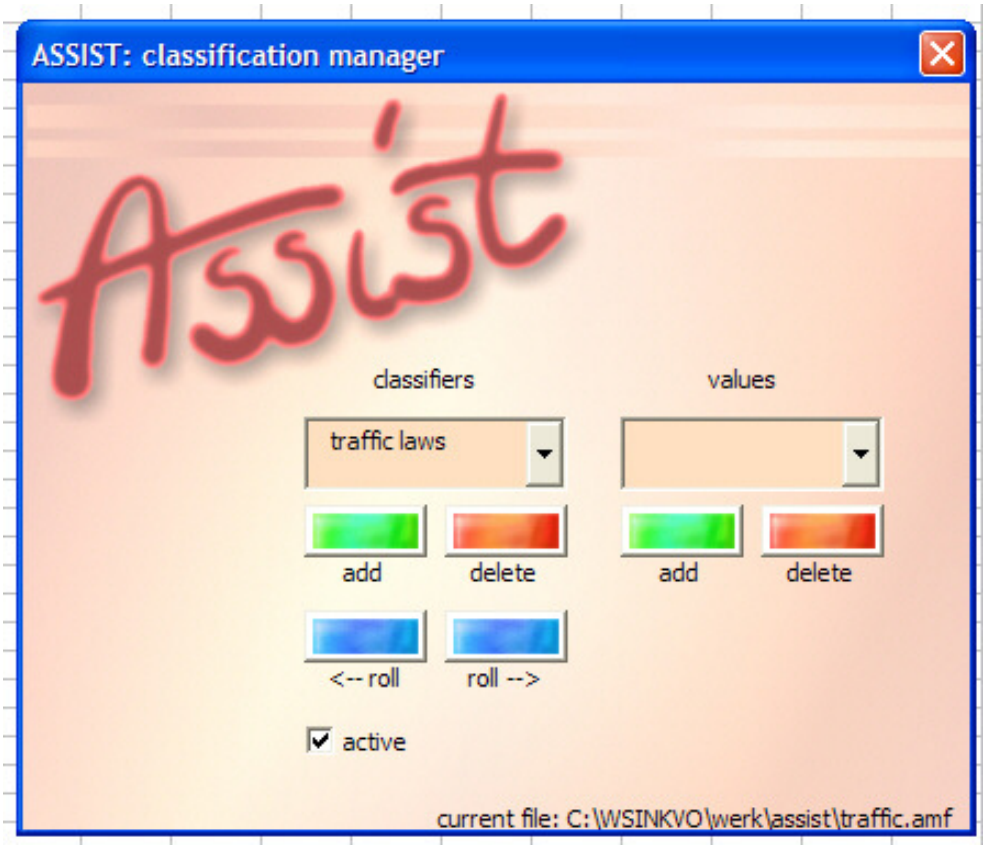


Figure 4

screenshot of the Classifier Manager-form

Controls

classifiers: this textbox-plus-pull-down-list contains the present list of classifiers. It serves to enter a new classifier (to be typed in the textbox), or to select an existing one. By clicking the down arrow button right of the textbox, we can see the entire list of presently existing classifiers. Typing a new text in the textbox allows to add a new classifier (by clicking the add-button underneath the classifiers-box.)

add: clicking the add-button underneath the classifiers box adds the text in the classifiers-textbox as a new classifier to the list of classifiers. Every new classifier automatically comes with one default value, indicated by '?'. Further values for this classifier can be added by typing values in the values-textbox and clicking the add-button underneath.

delete: this deletes the presently selected classifier *and all its values*.

<--roll: the order of the classifiers can be adjusted by the two roll-buttons. The button <--roll moves the presently selected classifier one place up in the list.

roll-->: this moves the presently selected classifier one place down in the list of classifiers.

active: by default, classifiers are active. That means that they are used to distinguish items; they are taken into account to see if classifiers are sufficiently independent (see below), they are used in clustering (see below) and they are used in the process of generating random new items where it is required that the combination of values of all *active* classifiers of the new item is unique. It can be meaningful to temporarily switch off a classifier, however. For instance, to get a better understanding of the group-structure (the *cluster* structure) of the present collection of items. This means that a non-active classifier is still depicted in the table, but its values are ignored by ASSIST in all processing. The name of an inactive classifier appears in the table with a diagonal cross through it (see the screenshot of the classifier-part of the item table below). The activity-status of a classifier can be toggled by clicking the checkbox for the currently selected classifier. There is an intimate connection between the activity-status of classifiers and clustering. The clustering tool contains sliders that are used to adjust the relative importance of classifiers in computing the distance between items; if such a slider is put to 0, this signifies that the associate classifier has no influence; in that case, the activity status of the classifier is set to 'not active', the name of the classifier is crossed out, and a new clustering is calculated where this classifier is ignored. Conversely, when the clustering tool is started, the sliders associated to classifiers with status 'not active' are set to zero by default. (See below for a further explanation of the clustering mechanism.)

values: this textbox-plus-pull-down-list contains the list of values for the presently selected classifier. By clicking the down arrow button right of the textbox, we can see the entire list of presently existing classifiers. Typing a new text in the textbox allows to add a new value for the present classifier (by clicking the add-button underneath the values-box.)

add: clicking the add-button underneath the values-textbox adds the text in the values-textbox as a new value to the list of values for the presently selected classifier.

delete: clicking delete underneath the values-box deletes the presently selected value.

Classify

General introduction

The Classify-form is used to build and use the ontology. It assumes that classifiers and their values have been entered by the Build Classifier-tool. With these classifiers, we can fill in, for every item in the item-list, the values for each of the classifiers. In this way, the world knowledge about the items is formally represented in ASSIST. Once this information has been represented in ASSIST, we can use this information in various ways:

- we can verify if the collection of classifiers is sufficient to distinguish all items
- we can check if the various classifiers are sufficiently orthogonal (independent) to form a meaningful representation of the underlying world knowledge
- we can automatically generate new combinations of classifier values that might inspire to new items (this is done using the produce random item-button on the Inspire-tool
- we can automatically cluster the items in the list. One form of clustering is provided by *lexicographic* sorting: we put the classifiers in a desired order, and next sort the items with respect to their value for the first classifier; if several have the same value for this classifier, these are sorted with respect to the second classifier, and so on – until we can assess if all items are distinguishable with the present set of classifiers or not. A more sophisticated form of clustering, that does not require a priori ordering of the classifiers will be implemented in the future.

Manual

The form looks as follows:

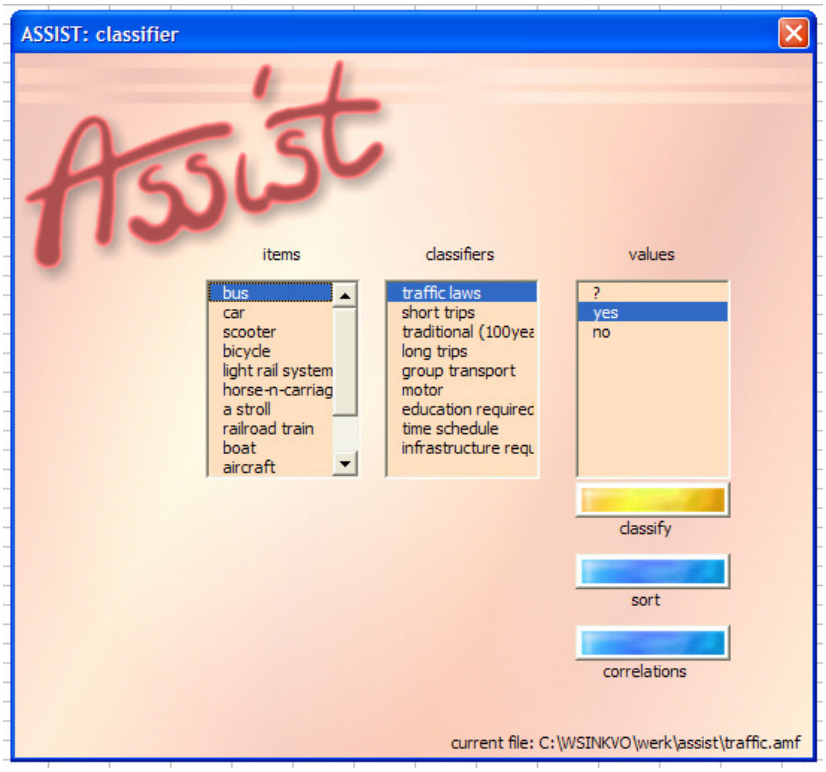


Figure 5

screenshot of the Classifier-form

Controls

items: this list contains the names of all present items. We can select the item we want to classify. If an item is selected, the value of that item for the presently selected classifier (as selected in the classifiers-list box) is automatically highlighted in the values-list box.

classifiers: this list contains the names of all presently existing classifiers. We can select the classifier we want to set the values for. If a classifier is selected, the list of its values appears automatically in the values-list box.

values: this list contains the values for the presently selected classifier. The value that is highlighted is the value as it applies to the presently selected item. If we want to classify an item with respect to a classifier, we select the appropriate value from the list values and click the classify-button.

classify: clicking the classify-button establishes a mapping between the presently selected item, the presently selected classifier and the presently selected value.

sort: this button performs two actions. First, it performs a lexicographical sort of all items with respect to the *active* classifiers in their present order. Next, it applies a colouring scheme to the concepts. Every row is coloured on the basis of the values of the classifiers in that row. If two subsequent rows are identical with respect to the values of the *active* classifiers, they receive the same colour, otherwise they are coloured differently. In other words: two subsequent rows with the same colours indicate two items that are indistinguishable as far as the present *active* classifiers are concerned. From the resulting colouring, it is immediately clear if groups or *clusters* of concepts exist that are indistinguishable with respect to their classifier values: they appear in single-coloured bands, whereas items that are distinguishable from all other items have a colour that differs from their successor and predecessor. Sorting can be used for two purposes:

- to verify if a present choice of classifiers is sufficient to represent all distinctions in the items domain;
- in case there are clusters of indistinguishable items, it shows which these clusters are, so when inventing a next classifier, the user can focus on the items that should be distinguished by means of this new classifier. Viewing the clustering structure is facilitated further by means of the clustering and visualization tool; see below.

Every time when classifiers have been added or removed, sorting should be done to update the information about whether or not the current set of classifiers is sufficient to distinguish all items. Sorting obviously will affect the order of the items; this ordering is propagated consistently to all other tools in ASSIST that deal with (lists of) items.

correlations: Every active classifier can be assessed in terms of its ability to single out concepts. For instance, a Boolean-valued classifier that scores 9 out of 10 cases 'true' and only once 'false' gives much less information than a classifier that gives a 5-5 score. Also for classifiers with more than 2 values, ASSIST can compute how well the classifier balances. Ideally, every value occurs exactly the same amount of times. In that case, the imbalance is said to be 0. The other extreme is the hypothetical case where a classifier only produces a single value (in which case it doesn't contain any information at all). In that case the imbalance is said to be 1.

For any combination of two classifiers, say V and W, we can express how *independent* they are. To this aim, we average the imbalance for classifier V for each of the values of W, weighted with the number of occurrences of these values. For instance, consider the following case:

	value W1	value W2
value V1	20	3
value V2	2	3
value V3	3	4

This table represents the classifier values for 20+2+3+3+3+4=35 concepts. The combination where classifier V takes the value V1 and classifier W takes the value W1 occurs 20 times, et cetera. For value W2 of classifier W, we see that V is reasonably balanced. In order to quantify this imbalance, we introduce a function of two vectors, called the *normalized dot product*, which can be interpreted as the cosine of the angle of the two vectors. This function is abbreviated as dot(,). In the case of 1-dimensional vectors (=numbers, say x and y), we define it as $2 x y / (x^2 + y^2)$, which can be seen to vary between –1 and + 1. So it nicely corresponds with our intuition of the cosine of an angle. With this definition, the imbalance of the list of scores (3,3,4) is defined as 1-cos(α) where α=(arccos(dot(3,3))+ arccos(dot(3,4))+ arccos(dot(4,3)))/3 = (0+ 0.28+0.28)/3=0.19, and the imbalance is 0.017. For value W1, however, the numbers of occurrence of values V1, V2, V3 of 20, 2, 3, respectively, cause a much larger imbalance: α = 1.013, so 1-cos(α)= 0.47. The number of occurrences of W1 is 20+2+3=25 and the number of occurrences of W2 is 3+3+4=10, so the weighted average of the imbalances is (25*0.47+10*0.017)/(25+10)=0.34.

Since a measure for independency should be commutative, we also compute the weighted average for the transpose case:

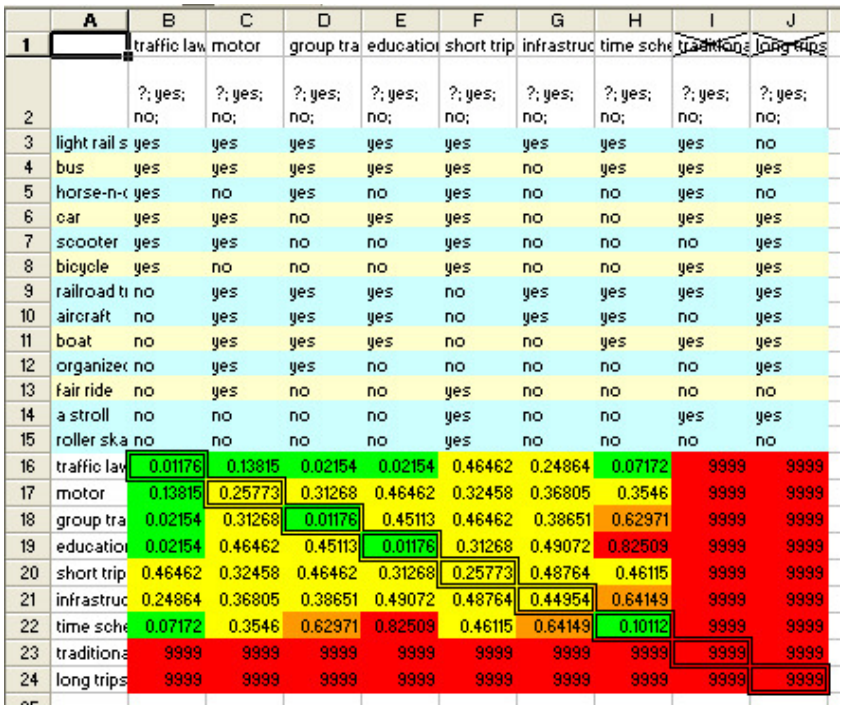
	value V1	value V2	value V3
value W1	20	2	3
value W2	3	3	4

With a similar procedure we find the value of 0.48 here. The final value for the independency of V and W is therefore (0.34+0.48)/2= 0.41. If this value is (close to) 0, the classifiers are (close to) independent.

Clicking the button correlations computes the imbalance for every active classifier and the independency for every pair of classifiers, and compiles this information in a (symmetric) matrix underneath the table, as follows (assume classifiers V, W, and X)

	V	W	X
V	imbalance of V	independency between V and W	independency between V and X
W	independency between V and W	imbalance of W	independency between X and W
X	independency between V and W	independency between X and W	imbalance of X

For an ideal case, where all classifiers are perfectly balanced and all pairs of classifiers are perfectly independent, all entries in he matrix are 0. In the worst case, entries are 1. For easier reference, cells are colour encoded: green means very good (very small values); yellow reasonably good; orange not so good and red means very large (very bad) values. Using the values in this matrix, we can define a 'quality' for a classifier, being the sum of the values in the column associated to that classifier. For instance, the quality of V is the sum of the imbalance of V plus the independency between V and W plus the independency between V and X. After having computed all imbalances and all independencies, correlations sorts all columns (all classifiers) in the order of decreasing quality. This is useful to help deciding which classifiers could probably be removed and for which better classifiers could possibly substituted. See the figure below for an example of the correlations matrix.



	A	B	C	D	E	F	G	H	I	J
1		traffic law	motor	group tra	educatio	short trip	infrastruc	time sche	tradition	long trips
2		?; yes; no;	?; yes; no;	?; yes; no;	?; yes; no;	?; yes; no;	?; yes; no;	?; yes; no;	?; yes; no;	?; yes; no;
3	light rail s	yes	yes	yes	yes	yes	yes	yes	yes	no
4	bus	yes	yes	yes	yes	yes	no	yes	yes	yes
5	horse-n-(	yes	no	yes	no	yes	no	no	yes	no
6	car	yes	yes	no	yes	yes	no	no	yes	yes
7	scooter	yes	yes	no	no	yes	no	no	no	yes
8	bicycle	yes	no	no	no	yes	no	no	yes	yes
9	railroad t	no	yes	yes	yes	no	yes	yes	yes	yes
10	aircraft	no	yes	yes	yes	no	yes	yes	no	yes
11	boat	no	yes	yes	yes	no	no	yes	yes	yes
12	organizec	no	yes	yes	no	no	no	no	no	yes
13	fair ride	no	yes	no	no	yes	no	no	no	no
14	a stroll	no	no	no	no	yes	no	no	yes	yes
15	roller ska	no	no	no	no	yes	no	no	no	no
16	traffic lav	0.01176	0.13815	0.02154	0.02154	0.46462	0.24864	0.07172	9999	9999
17	motor	0.13815	0.25773	0.31268	0.46462	0.32458	0.36805	0.3546	9999	9999
18	group tra	0.02154	0.31268	0.01176	0.45113	0.46462	0.38651	0.62971	9999	9999
19	educatio	0.02154	0.46462	0.45113	0.01176	0.31268	0.49072	0.82509	9999	9999
20	short trip	0.46462	0.32458	0.46462	0.31268	0.25773	0.48764	0.46115	9999	9999
21	infrastruc	0.24864	0.36805	0.38651	0.49072	0.48764	0.44954	0.64149	9999	9999
22	time sche	0.07172	0.3546	0.62971	0.82509	0.46115	0.64149	0.10112	9999	9999
23	traditiona	9999	9999	9999	9999	9999	9999	9999	9999	9999
24	long trips	9999	9999	9999	9999	9999	9999	9999	9999	9999

Figure 6

Example of the result of clicking sort and correlations. The coloring of items shows that 'a stroll' and 'roller skates' cannot be distinguished by the present set of classifiers and values, and neither can 'railroad train' and 'aircraft'; further, the correlation matrix shows that classifiers 'time schedule' and 'education required' are poorly distinguished. They are rather dependent: the color of the cells is red. The classifier 'traffic laws apply' balances well, which is indicated by the green color of the cell. Classifiers that are inactive are indicated by diagonal crosses; their matrix entries are indicated by the value '9999' and they are colored red. Correlation allows to distinguish 'good' classifiers (low values in the matrix) from 'bad ones'; after computing the correlations, ASSIST sorts the classifiers from good to bad; non-active classifiers therefore will always end up last.

Build Ranking System

General introduction

In the first phase of the innovation process, the purpose is to produce as many ideas (items) as possible; in the second phase, this multitude of ideas must be brought back to the very few very best ideas. This is done on in one of two ways, on the basis of *ranking*. The two ranking methods, supported by ASSIST are:

- Pareto front-ranking
- weighted sum-ranking

In Pareto front-ranking, we distinguish *dominated* from *non-dominated* items. A dominated item is an item for which another item exists that is better *in all respects.* For instance, suppose that the criteria we want to use for ranking are profitability and safety, both on a 100 point scale. An item with profitability = 50 and safety = 30 is dominated by an item with profitability = 60 and safety = 35. Such a dominated item should not be chosen. An item that is not dominated is called a non-dominated item; the collection of non-dominated items is said to form the *Pareto front*. Pareto front-ranking amounts to finding the Pareto front in a collection of items; those that are not on the Pareto front can be discarded.

Weighted sum-ranking assumes that each of the quality attributes i has a weight W_i. If the score of item j on attribute i is S_{ij}, the total score of item j is defined as $S_j=\sum_i W_i S_{ij}$. To distinct them from total scores, the values S_{ij} will be called *attribute* scores. The item with the highest total score is the "best". This is a bit naive, though: first of all, it is somewhat arbitrary to have precise numerical weights for quality attributes. This suggests much more precision than reasonably can be assumed. Suppose that profitability is more important than safety, should we then set $W_{safety} = 40$ $W_{profitability} = 45$, or $W_{safety} = 10$ $W_{profitability} = 90$? A similar problem applies to the attribute scores. Further, many attributes are not readily expressed as numbers: perhaps, profitability can be quantified more or less accurately on a numerical scale, but safety is much harder to cast in the form of a number. To cater for both difficulties, we have two mechanisms built in in ASSIST (a third, related mechanism is discussed in the description of the Auto Weights-form):

- All attribute scores S_{ij} , all total scores S_j , and all weights W_i are intervals with a lower border at least 0 and an upper border at most 100. The interval represents all possible numerical values for the weight or the score. The centroid of an interval is the "expectation value" of a weight or a score; the width of the interval indicates the (un)certainty. An interval I_1 is less than interval I_2 if the upper border of I_1 is less than the lower border of I_2. If the lower border of I_1 is larger than the upper border of I_2 we say that I_1 is larger than I_2. In all other cases, the order of I_1 and I_2 is not determined. So the weight [10:50] is not less than [30:70], even though its expectation value is less. Both weights are equally uncertain. Calculating with weights and scores as intervals protects us against hastily and non-substantiated conclusions: the total score S_j is only larger than an other total score $S_{j'}$ if the involved intervals, used to compute $\sum_i W_i S_{ij}$, are sufficiently narrow – that is: if we are sufficiently certain about a sufficient number of attribute scores and weights.
- Rather than assigning numerical values to lower and upper borders of intervals, we can input opinions about *relative* importance of quality attributes, and opinions about

relative values of attribute scores. Such opinions are called *quality rules*, and ASSIST can calculate sets of weights and attribute scores that comply with such quality rules. The underlying idea is, that in many cases it is easier to state quality rules than numerical values. A ranking process, employing quality rules, amounts to

- defining a sufficient number of quality attributes (to be called *attributes* for short; remember that the distinguishing aspects in the ontology were called *classifiers*, so a consistent nomenclature will avoid confusion between the (quality) attributes that are used in the ranking and the (classifying) attributes that form the ontology);
- entering opinions about the relative importance of quality attributes;
- entering opinions about the relative values of attribute scores;
- having ASSIST calculate a set of numerical intervals for the weights and attribute scores;
- checking if any items exist with a total score (interval) that is truly larger than the total score of *all* other items. In general, this will not be the case. We can then do one of the following:
  - try to narrow one or more of the intervals (because we *have* to make decisions about items that are going to be discarded, so we might as well be a bit bold in our opinions). Narrowing can be done by either scaling down all certainty intervals ('we are not certain, but let us pretend that we are a little bit more certain than we think'), or by making some of the opinions stronger by adding the predicate 'MUCH'. So we can say that A is better than B, whereas B is MUCH better than C. This is called an *enforced* opinion. ASSIST has a built-in numerical heuristics to deal both with regular opinions and with enforced opinions;
  - rather than focusing on one single "best" item we might continue working on a (small) collection candidates – similar to the Pareto front approach, where the selected collection also contains several items. The items to be chosen could, for instance, be those that are at least better than *some* of the other items. ASSIST can apply a colour coding scheme to label items that have total scores that are truly larger (or smaller) than any other items; these are coloured green (red). An item that is *both* truly larger and smaller than any other item is coloured yellow.
  - in some cases we have quality rules for some of the weights and/or some of the attribute scores, whereas we have numerical values for others. These two modes can be mixed. This is called *locking*: an attribute score interval can be locked to numerical values. This means that the calculation of actual weights and attribute scores on the basis of quality rules leaves the locked intervals untouched. Of course, it is possible that locking scores can conflict with quality rules; it is also possible that quality rules are conflicting. In those cases, ASSIST won't reach a consistent set of scores. It will signal this by reporting how many quality rules are violated.

The advantage of Pareto front-based ranking is, that we don't need the weights. With a different set of weights, the same Pareto front will result. A disadvantage, though, is that the number of non-dominated items can be prohibitively large (in particular if we have many attributes!) The weighted sum-ranking method can produce arbitrarily few solutions (if all the

interval widths are brought back to 0, there will, in general, be only one "best" solution – even though this may not mean a lot), but it has more room for arbitrariness.

ASSIST has been designed such that the two methods can be easily interchanged, depending on the characteristics of the present collection of items.

Several tools in the ASSIST tool suite are related to ranking. First, the Build Ranking System tool allows the definition of attributes, and it administrates quality rules. Next, the Rank tool allows to set and edit attribute scores and weights, and it offers control to the Pareto machinery and weighted sum calculations. Further, the Auto Weights tool controls the automatic calculation of weights from total scores and attribute scores. Finally, one of the visualization options in the Clustering and Visualization tool serves to visually inspect Pareto fronts.

Here we discuss the Build Ranking System-tool.

Manual

The form looks as follows:

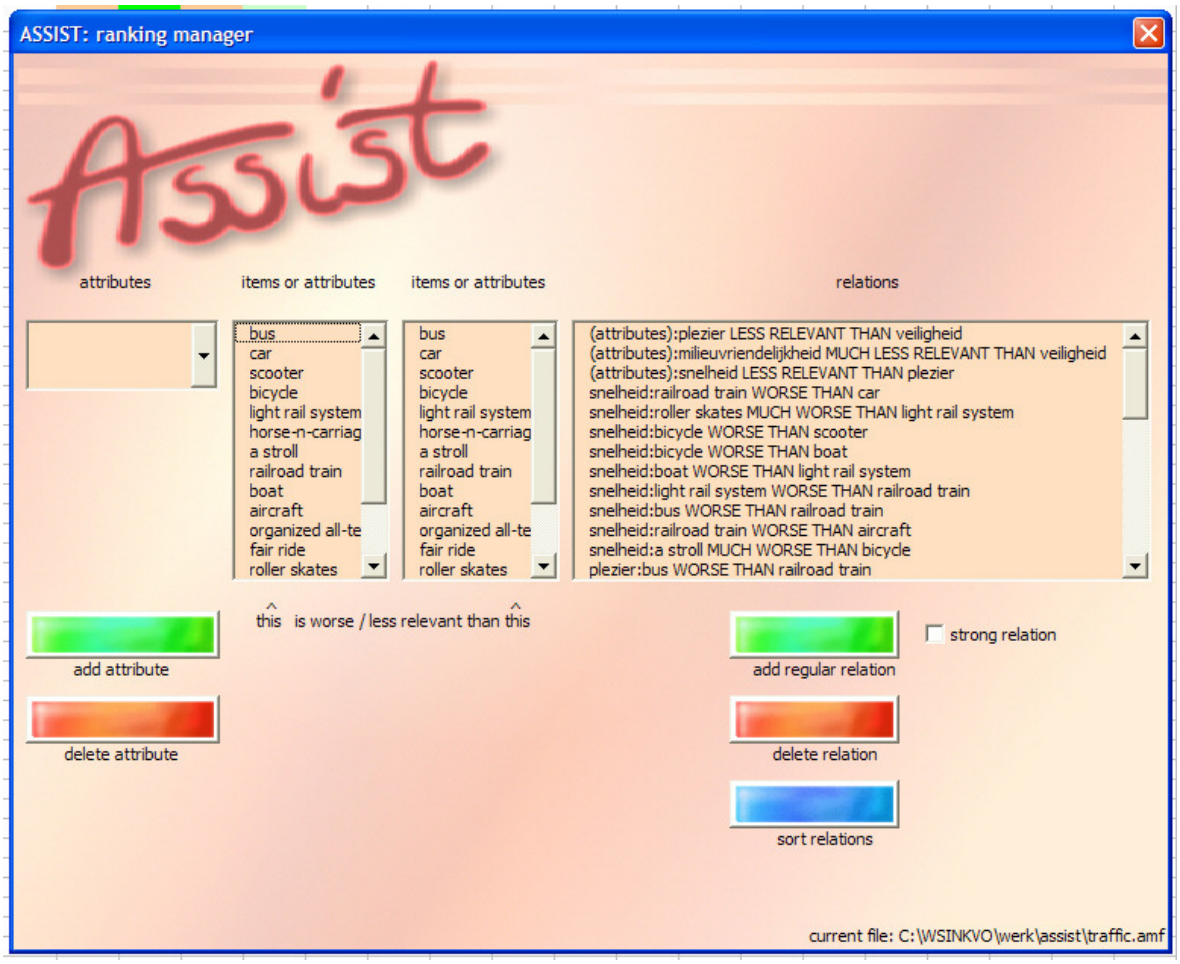


Figure 7

screenshot of the Ranking Manager-form

The left part of the form is devoted to administrating attributes; the middle part to quality rules. The existing quality rules are listed in the right hand part. Every quality rule is a statement that either gives the relative order of the (attribute) weights, or it gives the relative order of two attribute scores. Therefore, a quality rule requires two terms (both either attributes or items) that are selected in the two middle selection lists.

Controls

attributes: this combination-of-pull-down-list-and-textbox allows typing in a new attribute or selecting an attribute. An attribute is a text string. Attributes must have all distinct names. In order to create a new attribute, the text box should be made empty (if it isn't, select its contents and press 'del') and next a string should be typed; this string is added as an attribute by clicking the add attribute-button[2].

add attribute: this adds the string in the textbox in attributes as a new attribute.

delete attribute: the attribute that is selected in the selection list in attributes is deleted.

items or attributes (1) : this is a complete list of all presently existing items, plus all the presently existing attributes. An element of this list is to be selected if a new quality rule is to be made. The item (or attribute) in this, left most, list is worse (if it is an item) or less relevant (if it is an attribute) than the one selected in the right most list items or attributes (2). Quality rules can be about relative relevance of attributes, or they can be about attribute scores.

- If the quality rule is about the relative relevance of attributes, the term '(attributes)' should be selected in the list attributes.
- If the quality rule is about the relative quality of items with respect to a particular attribute (that is, if the quality rule is about attribute scores), the actual attribute is the one presently selected in the list attributes.

items or attributes (2) : this is the second complete list of all presently existing items, plus all the presently existing attributes. An element of this list is to be selected if a new quality rule is to be made. The item (or attribute) in this, right most, list is better (if it is an item) or more relevant (if it is an attribute) than the one selected in the left most list items or attributes (1).

relations: this list contains all the presently existing quality rules. Every quality rule comes in the form of a relation, either between two attributes (one attribute being less relevant than the other), or between two attribute scores (one attribute score being worse than the other). If we want to remove a relation, we can select the relation to be removed in this list. If a relation is enforced (using the check box, see below), it contains the word 'MUCH'. Further, if in the present situation a relation is violated, this is also indicated in the list box. However, violated relations are only indicated after a full calculation of the scores is performed, so the absence of a violation-warning does not guarantee that the relation is satisfied.

add relation: if this button is clicked, there should be an attribute (or the word '(attributes)') selected in the list attributes, and an item or attribute in the list items or attributes (1) and an item or attribute in the list items or attributes (2). With these three ingredients, a quality rule is formulated and added to the list of quality rules.

---

[2] Notice: in this screenshot, attributes are in Dutch. 'Snelheid' means 'velocity'; 'plezier' means 'fun'; 'milieuvriendelijkheid' means 'environmental friendliness', and 'veiligheid' means 'safety'.

strong relation: this check box is used to indicate if a relation should be enforced, or if an enforced relation should be de-emphasized. Checking or un-checking the check box affects the currently selected relation.

delete relation: the selected relation in the list relations is deleted

sort relations: quality rules can be added in arbitrary order. This means that the relations in the list relations can occur in random order. For a better overview, they can be sorted.

Rank

General introduction

The Rank-form is the most important form to be used during the ranking process. It allows to input and inspect weights and attribute scores; it controls ASSIST's automated computation of weights and attribute scores to comply with quality rules; it controls which of the two ranking methods is being used (Pareto front-ranking or weighted sum-ranking). Further it assists in narrowing or widening uncertainty intervals. The current states of all weights and attribute scores can be inspected in the Rank-form; it is easier, however, to see this information in the part of the table in worksheet "Items" that is devoted to the ranking (=the right most part of the table). The worksheet "Items" contains, in the form of a table, all attribute score intervals. For easier reference, these are colour-encoded. The convention is that colours between green and red indicate "high" values vs. "low" values. More saturated colours mean: smaller intervals (=more certainty); less saturated colours mean: larger interval (=less certainty). The interval [0:100], which is the largest uncertainty, corresponds to white. [0:0] is the lowest possible value with largest precision is bright red, and [100:100] which is the largest possible value with largest precision is bright green. This colouring scheme is used to colour weights, attribute scores, and total scores for items. The item names themselves are coloured red, green, yellow or white.

- In weighted sum-ranking, the meaning of the colours is as follows:
  - red: the item is worse than some other item;
  - green: the item is better than some other item;
  - yellow: the item is both worse than some item and better than some other item;
  - white: none of the above.
- In Pareto front-ranking, the meaning of the colours is:
  - green: the item is on the Pareto front: it is non-dominated;
  - white: the item is not on the Pareto front: it is dominated by some other item.

Also, the colour scheme is used to quickly set an attribute score or a weight. An example of the attribute-rank part of a table, including attribute scores and item colouring is depicted below.

K	L	M	N	O	P	G
	plezier	veiligheid	snelheid	milieuvriendelijkheid		
	[50:97]	[97:10(	[0:50]	[0:46]		
	[0:48]	[54:100]	[77:77]	[50:74]		
	[48:90]	[8:54]	[0:75]	[49:49]		
	[20:100]	[0:100]	[0:59]	[74:100]		
	[0:20]	[0:8]	[76:100]	[0:49]		
	[0:100]	[0:50]	[59:100]	[0:32]		
	[36:64]	[50:100]	[31:57]	[48:100]		
	[90:100]	[8:54]	[77:77]	[48:48]		
	[0:100]	[54:100]	[76:100]	[0:32]		
	[20:100]	[0:100]	[57:78]	[0:100]		
	[0:100]	[0:100]	[0:100]	[0:100]		
	[0:100]	[0:100]	[0:100]	[0:100]		
	[0:36]	[50:100]	[0:0]	[48:100]		
	[64:100]	[0:50]	[29:58]	[48:100]		
plezier	0.14762	0.24671	0.01356	0.1465		
veiligheid	0.24671	0.17469	0.29235	0.32155		
snelheid	0.01356	0.29235	0.16379	0.71214		
milieuvrie	0.1465	0.32155	0.71214	0.1336		

Figure 8

A color scheme is applied to the table with attribute scores

The top row contains the names of the attributes. In the second rwo, we see the attribute weights, in this case the intervals [50:97], [97:100], [0:50] and [0:46]. These are the weights for the attributes, 'plezier' (='fun'), 'veiligheid' (='safety'), 'snelheid' (='velocity') and 'milieuvriendelijkheid' (='environmental friendliness'), respectively. In subsequent rows we see all the item scores for all respective items. The lower most rows form a correlation matrix, in much the same sense as the correlation matrix we saw earlier for classifiers. The present correlation matrix, with the same colour interpretation as before, helps to assess the relative independency of attributes. Again attributes are sorted, from left to right, from 'good' (=relatively well-balanced attributes that are relatively independent from others) to 'bad' (=relatively poor-balanced attributes that are relatively dependent from others). In order to get the attribute correlations, the correlate attributes-button in the Rank-form has to be clicked (see below).

The total item scores can be seen in a graph, which is also found in the worksheet "Items", provided that the check box graph visible in the Rank-form is checked (see below). In this graph, every item corresponds with one horizontal bar. If there are not too many items, each bar is preceded by the name of an item; otherwise, we can find the item that corresponds to a bar by simply counting the bars, since the bars are in the same order as the items in the table. A bar is subdivided in three segments. The border between green and yellow is the lower border of the interval, corresponding to the total score of the item; the border between yellow and red indicates the upper boundary. In rare cases (intervals with width=0), the yellow bar vanishes.

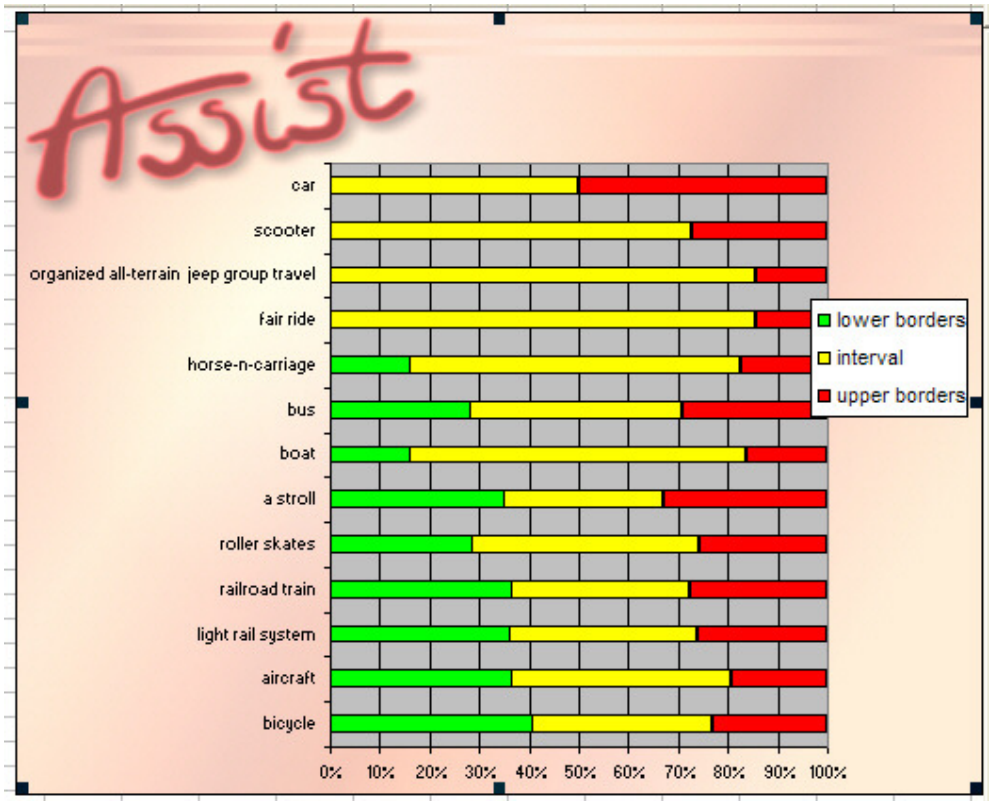


Figure 9

The total scores for the items are visualized with colored bars.

Manual

The form looks as follows:

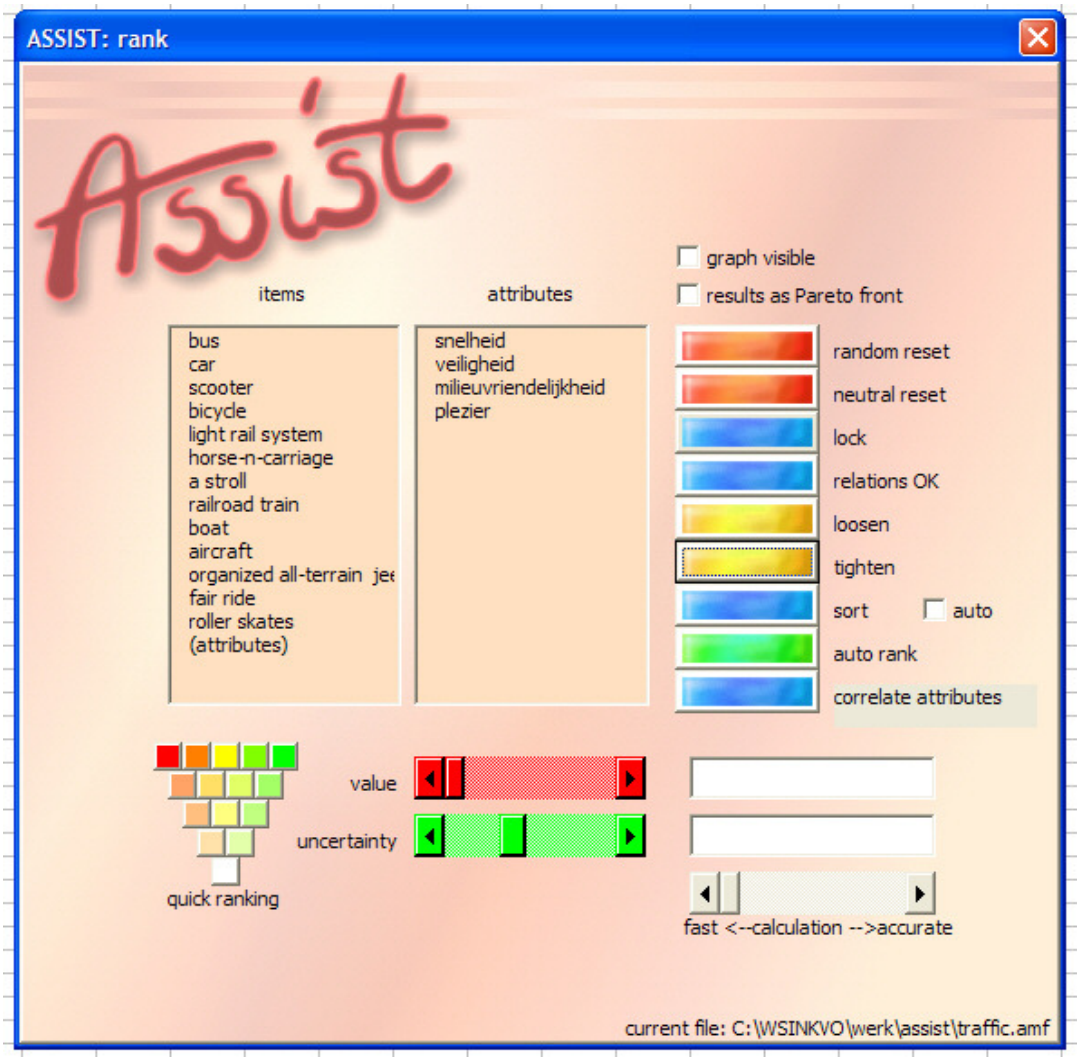


Figure 10

screenshot of the Rank-form

Control

items: this contains the list of presently existing items, plus the word '(attributes)'. If we want to set or inspect an attribute score for an item and an attribute, we select the item in this list. If we want to set or inspect the weight for an attribute, we select the word '(attributes)'. After selecting an item (or the word '(attributes)'), and an attribute, the present attribute score or attribute weight is automatically converted to the two sliders value and uncertainty. The numerical lower and upper boundaries are filled in in the text boxes right next to the sliders. The colours of the sliders and the textboxes represent the present interval values.

attributes: this contains the list of presently existing attributes. If we want to set or inspect an attribute score for an item and an attribute, we select the item in the items-list, and the attribute in this list. If we want to set or inspect the weight for an attribute, we select the entry '(attributes)' in the items-list and the attribute for which we want to set or inspect the weight in

the attributes list. After selecting an attribute, the present attribute score or attribute weight is automatically converted to the two sliders value and uncertainty. The numerical lower and upper boundaries are filled in in the text boxes right next to the sliders. The colours of the sliders and the textboxes represent the present interval values.

quick ranking: the cluster of 5+4+3+2+1=15 coloured buttons form a quick and intuitive way to set a value to an attribute score or an attribute weight. The 5 buttons in the above row correspond to the zero-uncertainty intervals [0:0], [25:25], [50:50], [75:75] and [100:100], respectively. The second row corresponds to the low-uncertainty intervals [0:40], [20:60], [40:80] and [60:100], respectively. The third row corresponds to the medium-uncertainty intervals [0:50], [25:75], and [50:100], respectively. The fourth row corresponds to the high-uncertainty intervals [0:66] and [33:100]. Finally, the fifth row corresponds to the maximal uncertainty interval [0:100]. Notice that, for more intuitive operation, the colours of the buttons represent the corresponding intervals. Clicking on of these buttons fills in the chosen interval, either for the present attribute score or for the present attribute weight. The sliders value and uncertainty are set to the appropriate values, the textboxes contain the numerical values for lower and upper bound, and the contents of the table and the graph are updated. The quick ranking devise is well suited for ranking items and attributes for which a merely intuitive score can be given. For those cases where more precise, numerical values exist, we can use the value and uncertainty sliders.

value: in cases that we have numerical evidence for an attribute score or attribute weight, we can input this by means of the sliders value and uncertainty. We should realize, however, that the lower and upper bounds for an interval are restricted between 0 and 100. This means that, for instance, a value of 20 cannot have a larger uncertainty than 20, and similar for a value of 80. The sliders won't admit a setting that corresponds to an interval outside the 0...100 range.

uncertainty: in order to set a desired value it may be necessary first to adjust the uncertainty slider in order to have the corresponding interval within the 0 ... 100 range.

graph visible: mark this checkbox in order to produce the graphical display from figure 9.

results as Pareto front: the colour scheme that is applied to the items in the table in worksheet "Items" can either be based on the Pareto front-ranking or on the weighted sum-ranking. If the checkbox results as Pareto front is checked, the items that are on the Pareto front are coloured green in the table; the other items are red. Otherwise, the white-yellow-green-red scheme is used as explained above. This colouring scheme in case of Pareto front-ranking is not affected by the values of the attribute weights. Nevertheless, the expressions for the item scores, $S_j = \sum_i W_i S_{ij}$, are also calculated in this case, and the graph shows the order of the items on the basis of the S_j even in the case results as Pareto front is checked.

random reset: clicking random reset assigns random values to all attribute scores and to all attribute weights, except those who have been locked (see below for an explanation of the locking mechanism). In order to get an idea of the reliability and accuracy of a ranking outcome, the following steps should be followed:

- first, assign attribute scores and attribute weights for those cases where a reasonably certainty exists, and lock these entries.
- next, add relations to represent the quality rules as far as they can be formulated.

- next, click random reset. As a result, most of the quality rules will be violated. This is indicated by the caption of the button press to fix ... relations: this will state how many relations (quality rules) have been violated.
- click the button press to fix ... relations; this starts an iterative process which gradually adjusts all attribute scores and all attribute weights in an attempt to satisfy the quality rules. It is possible that, given the random initial setting, that this process does not converge (this also depends on the setting of the slider fast<--calculation-->accurate). In that case, click again random reset and press to fix ... relations. If the process never converges, it is likely the case that the quality rules are contradicting[3].
- even if the process converges in a situation which reads 'relations OK' (which means that all quality rules are satisfied, it can be instructive to repeat the process a couple of times with different sets of random initial values. Indeed, a given set of quality rules usually allows a large amount of different sets of attribute scores and attribute weights, and in different configurations, different items may prove "the best"[4].

neutral reset: instead of random reset, one may also use a neutral initial situation. That is: all attribute scores and attribute weights, except the ones that are locked, are set to [0:100]. In this case, the iterative process to comply with the quality rules will strive for intervals that are as large (=as uncertain) as possible.

lock: an attribute score or an attribute weight may be locked. That is: the value will not be affected by a random reset or neutral reset, nor by clicking loosen or tighten, nor by clicking press to fix ... relations. Locked values will be affected though, by adjusting the sliders value and/or uncertainty, or by one of the quick ranking buttons. Values that are locked are indicated in the table by a double underline. If an item/attribute pair is selected in the items and/or attributes lists, the caption of the lock button changes in accordance to whether the presently selected attribute score or attribute weight was locked or not. If it was locked, it can only be unlocked; if it was not locked, it can only be locked.

press to fix X relations: the caption of the button press to fix ... relations indicates how many relations (quality rules), in the present situation, are violated. If no quality rules are violated, the caption says 'relations OK', so it is not necessary to click this button. Otherwise, clicking the button press to fix ... relations starts an iterative process that attempts to apply minimal adjustments to the present attribute weights and attribute scores in order to satisfy all the quality rules. The process stops after a number of iterations that should normally be enough to reach convergence, *if a solution exists*. This number of iterations is adjusted with the slider fast<--calculation-->accurate. If the porcess doesn't reach convergence, the caption will indicate how many unsatisfied relations remain. A second click (=a further round of iterations) may occasionally give a satisfying solution; more commonly, we have to attempt a different initial configuration (click random reset), or there is no solution at all because conflicting quality rules and/or locked attribute scores or attribute weights contradict the quality rules.

---

[3] Quality rules can easily be contradicting, for instance by demanding that A > B, B > C, and C > A, where A, B, and C are attribute scores or attribute weights. A contradiction may also occur if locked attribute scores or locked attribute weights conflict with some quality rules.

[4] If the slider fast<--calculation-->accurate is moved to the extreme right, the outcome of the calculation of the scores should normally be unique, provided that a solution exists (that is, that no conflicting rules have been entered) .

loosen: the amount of items for which the total score is truly better than for any other item depends on the width of the intervals. If all intervals are of maximal width (=maximal uncertainty), there will be no single item that is truly better than any other; if all interval widths are 0, there is most likely precisely one solution which is single best. To get a reasonable number of items, we may therefore adjust the widths of uncertainty intervals. This, of course, can be done on a per-interval basis by selecting an item-attribute pair and next using the sliders value and uncertainty. It is much quicker, though, to adjust all intervals simultaneously. Clicking the loosen-button increases all the intervals with 60% of the width. loosen and tighten do not apply to locked attribute scores or locked attribute weights. By applying loosen, intervals cannot become to extend the range 0...100. This means that repeated clicking on loosen may cause intervals to *saturate*, that is: to get a lower value of 0 or an upper value of 100. In that case, the effect of loosen can no longer be undone by *tighten*.

tighten: clicking tighten is the converse of clicking loosen. Notice that intervals cannot become smaller than 0, though. If an interval gets too small, the machine representation to accurately represent the interval width can be too small. In that case, the effect of tighten cannot exactly be undone by loosen. Normally, the tighten operation is done if there are too few items that are truly better than any other items, that is (in the weighted sum-ranking method), too few green items. In the Pareto front-ranking method, loosening and tightening intervals has no effect on the number of items that are on the Pareto front – unless, by repeated loosening, intervals reach the maximum uncertainty of [0:100].

sort: the main reason why total item scores are computed, is to find out which items are better than others – or even: which items are better than *all* others. Therefore it is convenient to sort items on their total score. Of course, scores are intervals, and sorting intervals is not directly possible. In ASSIST, we have chosen therefore to sort on the expectation value of the interval, that is: on the average between lower and upper border.

auto: by checking the auto-box next to the sort button, sorting takes place automatically after every update of any of the attribute scores or attribute weights. This will probably be the standard situation; for large numbers of items, however, or for slow computers this may render the ranking process uncomfortably slow. Notice that sorting affects the order of the items, so after we go back to any of the other tools (Inspire, Classify, ...) we there find the items in an adjusted order as well.

auto rank: the ontology and the ranking system are two sides of the ASSIST system. At first sight, they seem to serve two very distinct purposes: the ontology, implemented in the Classifier-form helps to systematically formulate a fragment of world knowledge which provides a "coordinate system" for the space of items. The Rank-form helps to select suitable items. There is a close connection between the two, though. By means of the ontology it is possible to define a *distance* between any two items. Indeed, items who have the same set of values for all classifiers cannot be distinguished; they are identical as far as the ontology is concerned (even though they might have different names: the Morning Star is the same object as the Evening Star, because they have the same bundle of attributes). We can use the number of classifiers in which two items differ as a simple (naive) measure for their distance. This distance is called the Hamming distance. In ASSIST we use a simple variation to the Hamming distance. Indeed, a classifier which can take a large amount of values can be said to have a different contribution to the distance than a classifier that has few different values. We adopt the convention that, the more different values a classifier can have, the smaller its contribution to the distance between two item. Once we know the distance (in the ontology)

between all pairs of items, we can use this distance to *predict* the result of ranking. Indeed, two items that have distance 0 should be identical in ranking. The smaller the distance between two items, the more similar their ranking values should be. This leads to the following idea: suppose we have a collection of items, T_i , i=1,2,3,..., having attribute scores S_{ij} for a particular attribute j . Next there is a further item, X, with (ontological) distances D_i to each of the items T_i. Then it is a reasonable prediction to say that the attribute score for item X is given by $S_{Xj} = (\sum_i S_{ij} / D_i)/(\sum_i 1 / D_i)$. This is the so-called Shepard interpolation formula[5]; other, more advanced formulas can be used as well. When we click the auto rank-button, ASSIST computes the above estimate for the presently selected attribute score.

correlate attributes: Similar as the correlation of classifiers in the Classify-form, we can perform a calculation to assess the relative independency of pairs of attributes, and the degree to which attributes balance. Again, as with the classifiers, we depict the results in the form of a colour-encoded matrix, and we sort the attributes from 'good' to 'bad' ones.

fast<--calculation-->accurate: this slider adjusts the number of iterations of the numerical process that calculates item-scores and attribute weights in order to meet with the relations. This numerical process works as follows: all values (both lower bounds and upper bounds of all scores) are considered as point-masses, some of which are pair wise connected via springs. All the required effects are modelled by appropriate springs:

1. intervals should be as wide as possible: there is a pushing spring with rest length 100 between lower and upper border of each interval;
2. lower borders should not be less than 0: there is a conditional spring (= a spring which only comes into action if this condition is violated) on every lower border with zero-rest length and one fixed end in the value 0;
3. upper borders should not be larger than 100: there is a conditional spring (= a spring which only comes into action if this condition is violated) on every upper border with zero-rest length and one fixed end in the value 100;
4. for each interval, lower borders should be lower than upper borders: there is a conditional spring between lower and upper border of every interval which ensures the right order;
5. of a relation, the upper border of the lower interval should not be larger than the lower border of the upper interval: there is a conditional spring between these two borders that keeps them in the right order;
6. for every enforced relation, there is a virtual interval that should be as large as possible between the upper border of the lower interval and the lower border of the higher interval;
7. finally, for every locked interval, there are two zero-length springs that pull the two interval borders to the assigned values.

Springs 1 and 6 express conditions that should be optimised; all other springs express constraints that need to be satisfied. ASSIST implements a dynamical simulation of this spring-mass system, where all springs are slightly damped (to enforce convergence in a situation where all values come to rest); the springs of types 1 and 6 gradually get less stiff, and the springs 2, 3, 4, 5, and 7 gradually get stiffer during the iterative process.

[5] In this form, the interpolation is singular for D_i is 0. In that case, however, S_{Xj}=S_{ij}.

Auto Weights

General introduction

Weights can be automatically assessed (with come restrictions). Suppose that we have a number of items, and we can provide attribute scores for all these items for all (quality) attributes, but we find it difficult to give numerical values to the attribute weights. If we *can* give total scores for the items, however, ASSIST can attempt to calculate weights that are compatible with these total scores – provided that a set of weights exists that is compatible with the given attribute scores and total scores. This amounts to solving for the W_i from a set of equations $S_j=\sum_i W_i S_{ij}$. Since the set of items (=the set of j-s) is typically (much) larger than the set of attributes (=the set of i-s), these equations can only be solved exactly in rare occasions. If the attribute scores are sufficiently consistent, however, a so-called *least-squares optimal* solution will be generally quite reasonable. ASSIST applies an iterative solution strategy to approximate such a least-squares optimal solution. Automated calculation of weights is governed by the Auto Weights-tool.

Manual

The form looks as follows:

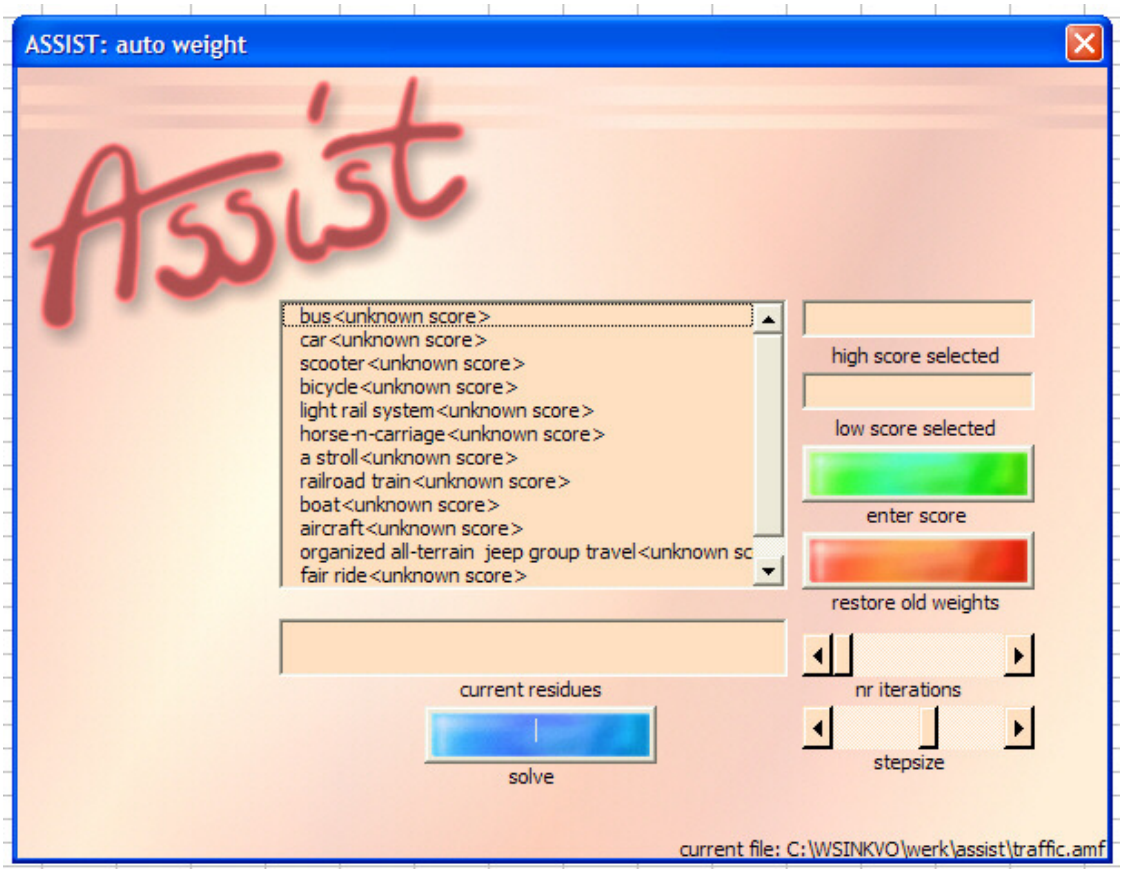


Figure 11

screenshot of the Auto Weight-form

Controls

main item list: this scrollable list contains all presently existing items. In order to set the total (item) score for an item, we select one in this list. Notice: not all items need to have a total score to be set.

high score selected: in this textbox we can type a number which is the upper value of the interval that represents the total item score for the presently selected item.

low score selected: in this textbox we can type a number which is the lower value of the interval that represents the total item score for the presently selected item.

enter score: once a lower and an upper score gave been entered, and an item has been selected, we can click this button; this sets the desired total item score for this item.

text box current residues: the iterative solver attempts to compute weights W_i such that $S_j=\sum_i W_i S_{ij}$ or $S_j - \sum_i W_i S_{ij} = R_j$, where the so-called residue R_j needs to be as small as possible. During the iteration process, the value of the residue is printed in the textbox labelled 'current residues', so that we can follow the convergence process.

solve: clicking solve starts the iteration process; it performs a number of iterations set by the slider nr iterations with step size given by stepsize. If convergence is not reached (that is, the residues as shown in current residues are still (too) large, we can click for another round of iterations.
restore old weights: in some cases, the process of automatically finding weights doesn't produce any satisfying weights (for instance if the attribute scores are not very coherent). In that case we may wish to go back to the initial attribute weights; we do so by clicking the restore old weights-button. (Notice: this is a sort of 'undo' function. It only applies to one round of clicking the solve-button. To make sure old weights can be restored for more extensive experiments it is therefore advisable to write the project to disk under a different name.)

nr iterations: this slider sets the number of iterations in one turn of clicking solve.

stepsize**:** this slider sets the step size. A step size that is too small causes slow convergence; a step size that is too large may cause instabilities.

Clustering and Visualization

General introduction

To get a better understanding of the structure of the set of items, two additional tools are provided within the ASSIST system. The first one is *clustering*; the second one is a visualization of Pareto fronts.

Clustering:

Between any two items, a *distance* can be defined. Such a distance should be zero for items that are identical; it should be non-negative, it should be symmetrical (distance between A and B equals the distance between B and A), and it should obey the so-called triangle inequality (the distance between A and B cannot be larger than the sum of the distances between A and C and between C and B). A suitable definition of distance as far as classifiers as concerned is the *Hamming distance*. The Hamming distance between two items is based on the number of classifiers in which these items differ. A classifier contributes to the distance inversely proportional to the number of values this classifier can take: if two items differ with respect to a binary classifier, this is more significant than when they differ with respect to an n-ary classifier, n>2. As far as the attributes are concerned, the distance between items is defined to be proportional to the difference between the expectation values (=the average between the lower and upper values of the interval). So for any two items we can compute a distance, which is a contribution of the distances due to classifiers and due to attributes. This distance is a (non-negative) number, which can be interpreted as the rest length of a spring. In this view, all items are point masses, connected via springs. Items that are close together are connected via short springs; items that have large distance have long springs between them. When we animate this network of mass points and springs, we get an intuitive view of the cluster-structure among the items. This is enhanced by the option to vary the influence of the various classifiers and attributes. For instance, if we fully ignore a particular classifier C, items will belong to the same cluster irrespective of whether they are equal or not with respect to C. The more classifiers or attributes we ignore, the fewer clusters will be formed. To see this effect, we associate a slider to each classifier and to each attribute. The sum of all contributions, that is: the sum of the values of all sliders equals 100%; increasing one slider (=increasing the influence of one classifier or attribute) will reduce the influence of all others. If the influence of a particular slider is set to 0, the associated classifier will be set to non-active, and the top part of the table of figure 6 will be re-calculated. This means that we see the clusters reflected both in the animation of the mass-spring system (as in figure 13) and in the blue-yellow regions in figure 6. Notice that attributes can not be set to non-active; however, in practice we will have only few non-zero sliders at any time; in most cases, the contributions to the distances of the attributes will be zero (since the Pareto plots offer more intuitive means to visualise the items with respect to their attribute values.) For this reason, the sliders for the attributes have been put on a different user interface page in the Cluster and Visualization form.

Pareto visualization:

For any two attributes, we can plot the graph of all items in a Cartesian coordinate system spanned by these two attributes. Every item is a dot in this graph; the uncertainty intervals give error bars to this dot. In case horizontal and vertical attributes are equal, all dots are on a (45 degree) line, and their order and spacing show the individual item scores. Such Pareto

plots are very useful to get an immediate understanding of trade-offs (with respect to any two attributes) between items.

The Clustering and Visualization tool provides controls for both kinds of plots.

Manual

First we explain the control of the animated mass-spring system to visualize the clustering. This system is operated by the buttons start animation, stir it up, and (optionally) stop animation. The two tabs classifiers and attributes allow access to the sliders to control relative influences of the various classifiers and attributes.

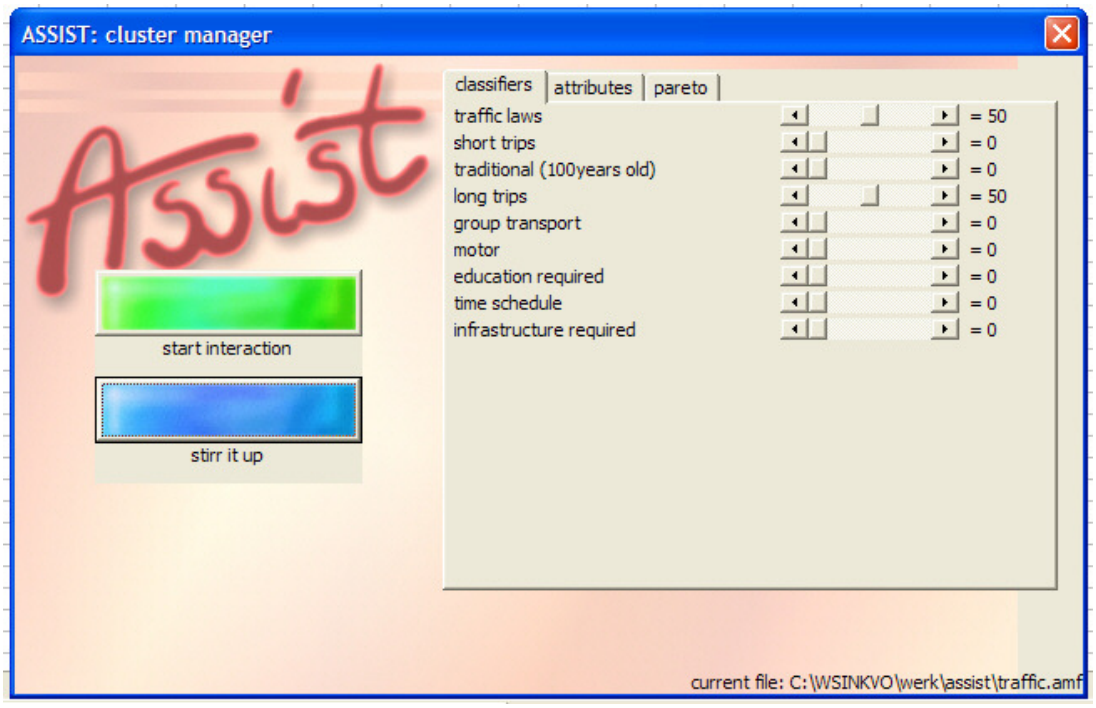


Figure 12

A screen shot of the Clustering and Visualization form for controlling the impact of classifiers on the clustering visualization. (A similar page is provided for controlling the impact of the attributes)

Controls

start animation: this starts the animation of the mass spring system. When the animation runs, all PC resources are consumed, and the other processes can get prohibitively slow. Therefore, the animation automatically switches off if no changes in any of the sliders has taken place for a certain amount of time. The animation can be stopped as well by clicking the stop animation button that appears as soon as the animation runs. During a running animation, the sliders can be operated freely in order to study the clustering structure. Figure 13 shows a screenshot of the animation. Notice that all items are identified by differently coloured markers. If a slider is set to 0, and a re-calculation of figure 6 takes place, the items may get re-ordered. In this case, the mapping between item names and markers may also get re-ordered. In order to get a better view to most of the markers, they are slightly offset in a random direction. So even if the distance between two items is 0, they appear at slightly different locations.

stir it up: in order to further facilitate the understanding of the clustering structure, we can re-start the animation from a random position. Even though the spring lengths stay the same (since these are calculated from the distances between the items), the configuration may look different. At least since the overall orientation of the mass-spring network is not defined by the spring lengths; further, its centre may be at random locations. (In rare occasions, few markers may end up outside the plot area. Re-clicking the stir it up button will probably put them back.

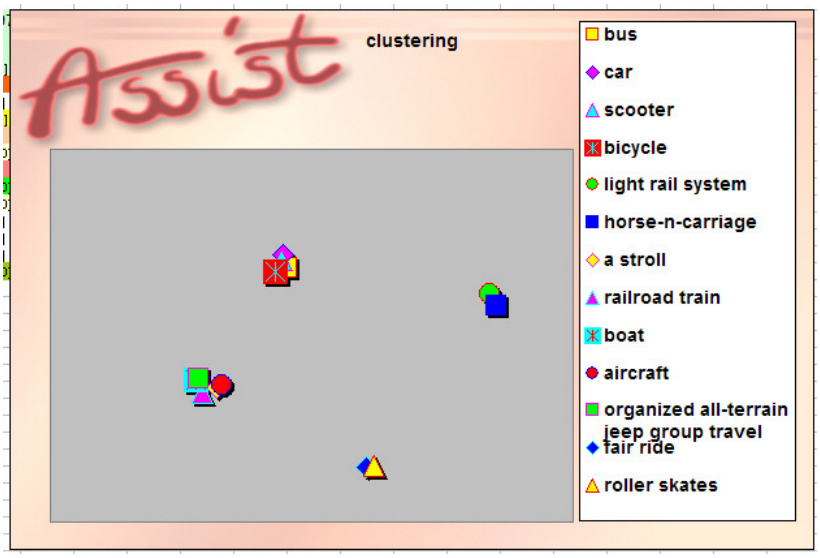


Figure 13

A still from a clustering-visualization

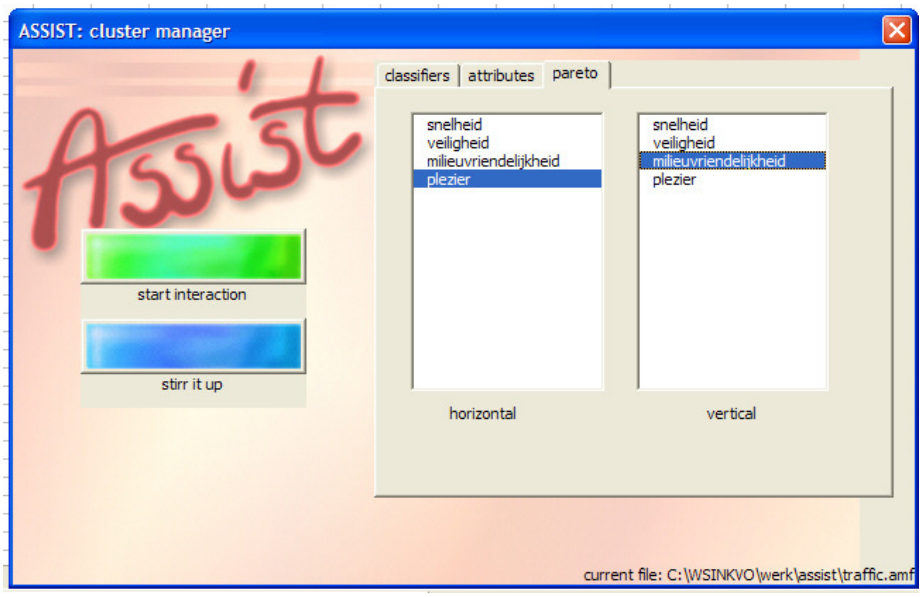


Figure 14

A screen shot of the Clustering and Visualization form for showing a Pareto front)

A second set of controls is provided to produce the Pareto plot visualizations. The buttons start animation and stir it up have no effect now; in fact, clicking the start animation button would stop the Pareto-visualization and re-start the visualization of the mass-spring system.

The generation of Pareto plots is controlled by clicking on two attributes, one in each list box (labeled 'horizontal' and 'vertical', respectively) of the Pareto tab in the Clustering and Visualization form. Clicking the first attribute constructs the actual Pareto graph; clicking the second attribute fills in the two axis labels and arranges all items in the graph. Again, items are indicated by colored markers; the legend contains these markers and gives the associated item names. All markers have error bars that indicate, in both attributes, the uncertainties.

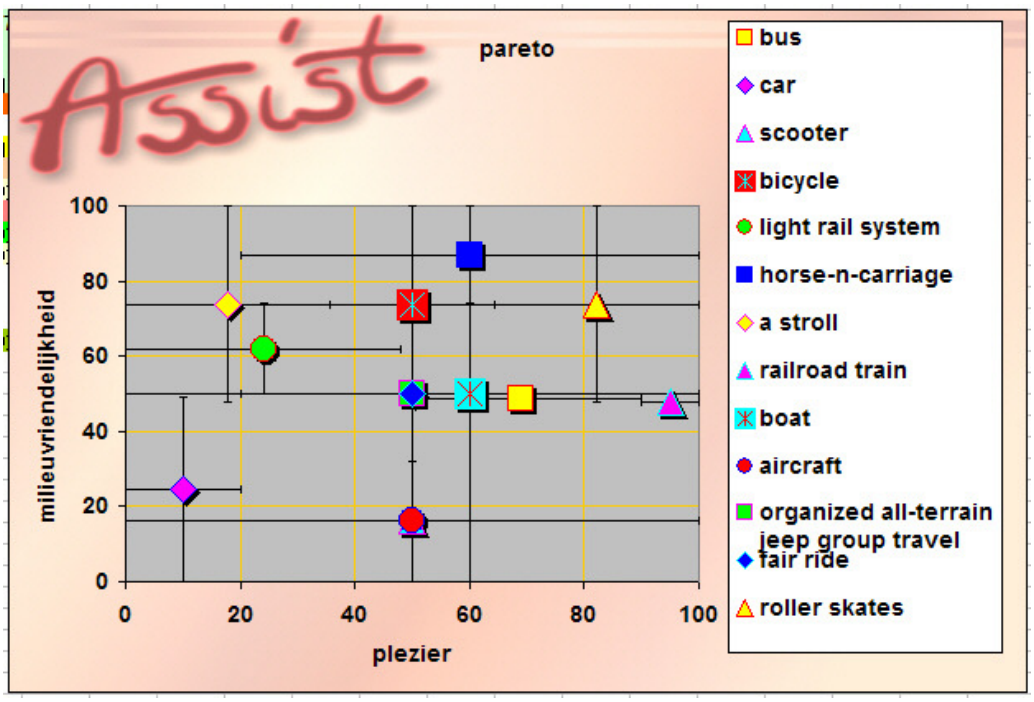


Figure 15

An example of a Pareto front visualization