Journal of Computer-Aided Molecular Design (2005) 19: 821–833
DOI 10.1007/s10822-005-9029-8

Automated QSPR through Competitive Workflow

J. Cartmell, S. Enoch, D. Krstajic & D. E. Leahy*
Cyprotex PLC, 13-15 Beech Lane, SK10 2DR, Macclesfield, Cheshire, UK

Received 6 June 2005; accepted 9 November 2005

Key words: ADME, autonomous agents, competitive workflow, molecular design automation, P-gp efflux, plasma protein binding, QSPR, solubility

Summary

This paper describes a novel software architecture, Competitive Workflow, which implements workflow as a distributed and competitive multi-agent system. The implementation of a competitive workflow architecture designed to model important computer-aided molecular design workflows, the Discovery Bus, is described. QSPR modelling results for three example ADME datasets, for solubility, human plasma protein binding and P-glycoprotein substrates using an autonomous QSPR modelling workflow implemented on the Discovery Bus are presented. The autonomous QSPR system allows exhaustive exploration of descriptor and model space, automated model validation and continuous updating as new data and methods are made available. Prediction of properties of novel structures by an ensemble of models is also a feature of the system.

Introduction

In this paper we introduce *Competitive Workflow*, a novel software architecture, implemented using autonomous software agents, for the automation of computer-aided molecular design and decision making processes. An implementation of competitive workflow, the *Discovery Bus* is exemplified for QSPR model building and prediction strategies. Results demonstrate how global models for some common ADME properties equivalent to those previously found by human experts can be derived automatically without expert intervention. These models are continuously updated as new data or new methods are implemented and the process exhaustively explores all combinations of modelling pathways, limited only by the available resources, principally CPU time.

*To whom correspondence should be addressed. Phone: +44-1625-505114; Fax: +44-1625-505199; E-mail: d.leahy@cyprotex.com

Automating problem solving strategies in drug discovery

Drug Discovery, like any research process, is characterised by hypothesis, search, experiment and backtracking, leading to the aggregation of knowledge. The Discovery path is not specified in advance but is created dynamically in response to changing goals, new data and evolving strategies and methods. Discovery takes place in a community of specialists, each of whom has available sets of component methods that can be assembled into one or more problem solving strategies. Specialists collaborate by providing solutions to each other, which when combined, create progress towards their shared goals. Although each specialist may only deploy a limited number of strategies, there are usually multiple instances of the components of a strategy, resulting in a combinatorial expansion in the number of instances of strategies.

The competitive workflow architecture enables implementation in software of the multiple specialist problem solving processes described above. It facilitates structural evolution in specialist capabilities, such as the introduction of new components and strategies as well as a continual reappraisal and adjustment to newly acquired knowledge such as the arrival of new screening data, new chemical source databases, synthetic methods or competitor patents. The architecture positively supports competition arising from the availability of multiple algorithms, strategies and data sources.

The most obvious metaphor for automation of problem solving strategies, is that of workflow, for which multiple patterns and commercial implementations exist [1, 2]. Applications of workflow have been available for a long time and there are many commercial toolkits available, with at least one available for cheminformatics (www.scitegic.com). There are also open source alternatives [3] under development. In addition, the workflow concept is central to many developments around the use of Web and Grid Services [4, 5].

The use of autonomous collaborating software agents to handle the complexity and dynamics required by some workflow management applications is an area of active research [6–8] and autonomous software agents have been applied to distributed learning [9], tool-integration [10], bioinformatics workflows [11] and systems biology modelling [12].

Studies aimed at the automation of the QSPR modelling process to compare the performance of descriptors [13] and both descriptors and correlation methods [14] have also been carried out previously.

The *Competitive Workflow* architecture described here takes advantage of concepts from both workflow and multi-agent systems [15] and has significant additional advantages. In particular, all possible combinations of components are explored leading to exhaustive evaluation of potential solutions. Since requests are always regarded as questions to which there are multiple answers and no request is ever regarded as complete, improved results emerge naturally as new strategies, components or additional data are made available.

The Discovery Bus therefore has the following distinguishing features: different technologies naturally compete with each other (and with any human agent), all combinations of components available are explored, and, results are automatically updated as new components become available. In conditions where there are insufficient resources to explore all possible combinations of strategies, which we expect to be the norm as more agents are added, then the most promising pathways are explored first by a reinforcement mechanism. Pathways leading to successful results are reinforced through increasing priority for CPU resource, creating a mechanism by which the community of agents can learn from experience.

As a consequence of these features, the Discovery Bus provides a mechanism for implementing knowledge acquisition and use in a highly efficient, comprehensive and continuously improving manner.

Application to QSPR

High throughput screening for common ADMET properties [16–18] creates an opportunity for the creation of QSPR models for routine selection of compounds for ADMET properties [19] as well as potency. However, constructing a QSPR experiment can be as difficult as designing many laboratory experiments, as there are a number of important choices that can drastically affect the quality of the resulting models [20]. Generally three key choices need to be made:

1. What type of model building technique is to be employed?
2. What type of molecular descriptors should be used?
3. How should the most appropriate set of the descriptors be selected to ensure an uncorrelated feature set is used?

Given the multiplicity of choices facing the QSPR modeller and the manual nature of modelling, QSPR model development can be labour intensive and re-work of an existing model, every time new data becomes available, can be impractical when there is limited modelling resource.

With the goal of extracting maximum value from the experimental generation of ADME data in mind, the Discovery Bus has been used to create an autonomous, continuously updated, exhaustive search QSPR modelling process, closely integrated with the production of experimental data. Automated exploration of available machine learning techniques for building QSPR models, molecular descriptor selection, and dealing with the problem of correlation between descriptors for several common ADME properties is described.

Computational methods

The competitive workflow architecture is implemented with multiple autonomous software agents distributed across a grid of servers, and these together comprise the Discovery Bus. The architecture shown in Figure 1 consists of classes of agents, with each class being able to respond to a specific type of request. In addition, there is a central planning agent, which captures the workflow underlying the specific problem strategy and which is responsible for receiving and processing data, by submitting requests for processing of subtasks.

As the input data becomes available for a specific execution the planning agent offers the execution to the worker agents on the worker servers via the dispatch agent. The worker agents only respond to executions which they can complete. Once the agent has carried out the execution its response is stored in the discovery state database.

In addition to these worker agents, there are also utility agents on the worker servers, such as Installer and Reaper. The Installer agent is responsible for installing new worker agents, whilst the Reaper agent automatically removes executions which have failed. The Controller and Scheduler are responsible for execution priority, which can be set manually, should results from specific agents be required during testing.

Agents respond to requests with results and new requests placed on the Discovery Bus, where classes of agents with appropriate capabilities respond. The process of offering data to classes of agents, and receiving responses from them continues until all available agents have responded to requests that match their capabilities. Since the requests remain on the Discovery Bus, if a new agent (such as a new descriptor or machine learning method) is implemented, the new agent responds and generates new requests in its turn. As a result, the new agent is automatically applied to all existing modelling activities.

One of the key advantages of the competitive workflow architecture detailed above is the ease in which additional functionality can be added to any of the classes of agents. For example, should a new classification method be required it is a simple matter of implementing the algorithm (or wrapping an existing program) and then installing it within the appropriate class of agent. Agents are implemented as shell scripts and are not limited to any particular technology, enabling easy integration of methods from multiple sources. Since all agent-generated requests are left active on the Discovery Bus, the new agent responds, and

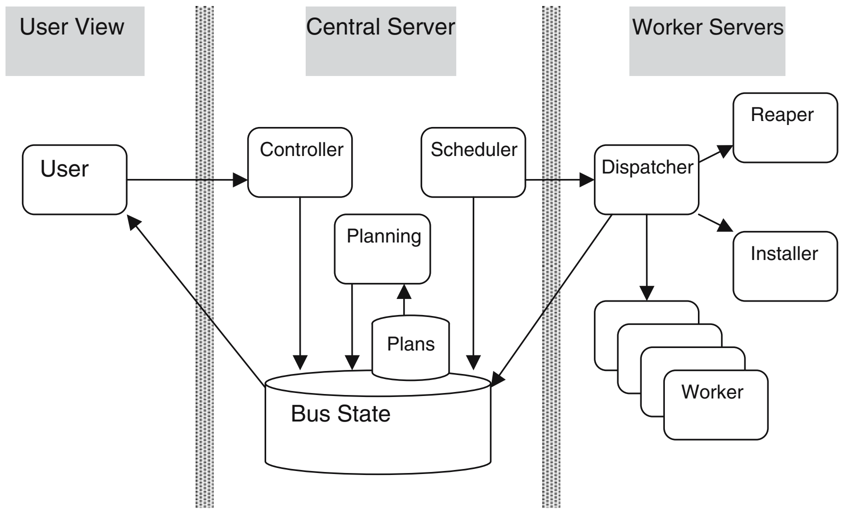

Figure 1. Distributed Multi-Agent Architecture of the Discovery Bus.

historical analyses are automatically re-evaluated using the new agent. The Discovery Bus maintains state through the use of an underlying database. A request to the controller causes the planning agent to retrieve the plan for that request type from the plans database. A plan is a process description that models a problem solving strategy as a competitive workflow. The executions within a plan are carried out asynchronously but sequentially with the output of previous executions being used as input for the next. Responses to requests from the planning agent can either be from individual agents responsible for a given task or other planner agents. Each execution has one or more output responses, and when the appropriate response has returned, the planner agent generates a request corresponding to the next step in the process. The software kernel is responsible for storing request and response data in the database and for delivery of this data on demand to the planner agent. The kernel also ensures that the agent executes only when the required input data is available and there is no repetition.

In the case of a QSPR model building and prediction workflow, the classes of agents include calculation of molecular descriptor, feature selection, data stratification and machine learning.

Molecular descriptors

This class of agents act on structural information, with each agent in the class calculating a different set of molecular descriptors. Currently over 2000 molecular descriptors are calculated by six agents implementing the following descriptor sets: ACD [21], Dragon [22], E-State [23], HE-State indices [23], LSER [24] and RECON [25]. We reference these as sets A, D, E, H, L and R respectively. Each agent responds by returning its descriptor set to the Discovery Bus. A descriptor combination stage is responsible for combining the individual descriptor sets. Various combinations of descriptor sets are produced (A&L, A&R&E, H&R&E, etc.) and these are returned to the Bus to be taken through the next stages (Figure 2).

Data stratification

Prior to any model building exercise the input data is stratified, that is divided into training and test data sets, with the training data set being used to build the QSPR models whilst the test data set is only used to assess the predictive power of the resulting models. The test data set is never used in the model building or selection process. The Discovery Bus implements the process of data stratification via a 'Stratify Data' agent, this agents algorithm functions as follows:

1. Order by property value the entire input data.
2. Select every 10th data point as belonging to the test data set.
3. All other data points belong to the training data set.

The training data set is then separated into 10 cross-validation sets using a similar algorithm in which for example, cross-validation set one contains compounds 1, 11, 21...; cross-validation set two containing compounds 2, 12, 22 and so on until no more compounds remain to be assigned.

By ordering the input data by property value the algorithm ensures that the test data set consists of a representative sample of the entire input data.

Feature selection

The CFS method of Hall [26, 27], to which we have added several variations, has been implemented. The agent is called once for every set of descriptors called by the combine descriptor process. Each time the agent is invoked it produces up to five responses, each of which is a different filtering of its input. The first two filters are always applied, (i) C-filter, where the descriptors are selected according to the core CFS method [26] and (ii) H-filter which is an extension to the core CFS method [27]. This method uses the core CFS algorithm, but with features added if their correlation with the target class is higher than the maximum correlation with features already accepted. The algorithm adds the features in the order defined by ranking of the difference of the two algorithms specified by Hall, the correlation with the target class and the highest correlation with a feature already accepted. Depending on the H-filter response, three responses are produced by further removal of the lowest ranked descriptors, (iii) Hr2-filter where H is reduced to contain half as many descriptors as there are data points, (iv) Hr4-filter where H is reduced to contain one quarter as many descriptors as there are data

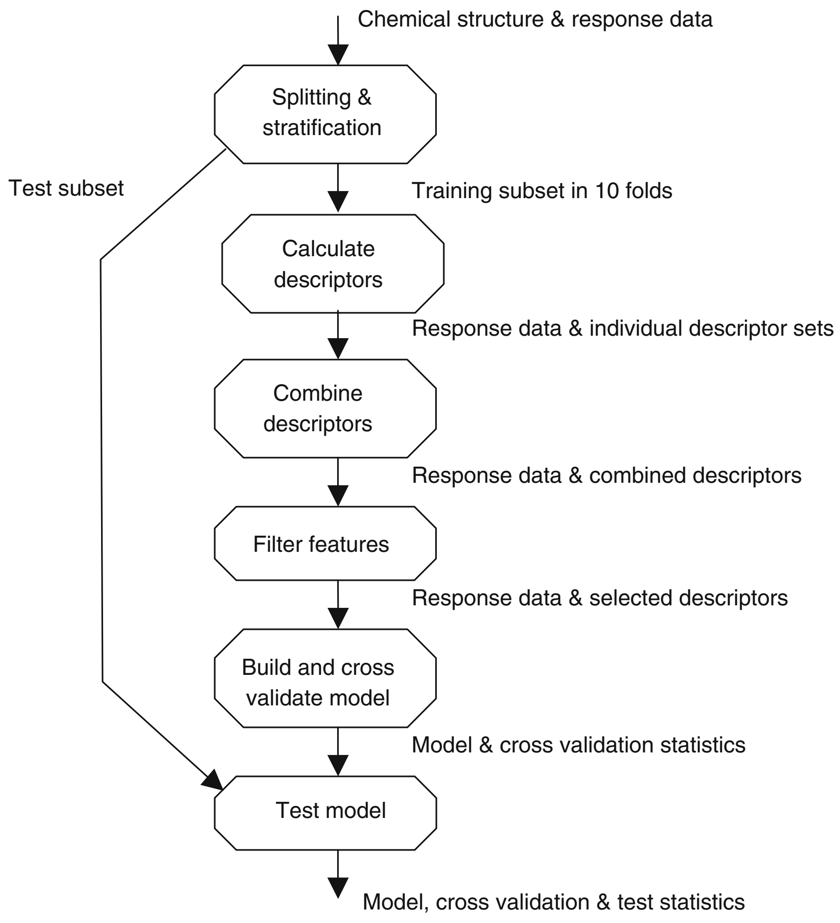
Chemical structure & response data
Splitting & stratification
Test subset
Training subset in 10 folds
Calculate descriptors
Response data & individual descriptor sets
Combine descriptors
Response data & combined descriptors
Filter features
Response data & selected descriptors
Build and cross validate model
Model & cross validation statistics
Test model
Model, cross validation & test statistics

Figure 2. Overview of the main stages of QSPR model building.

points, and Hr10-filter where H is reduced to contain one-tenth as many descriptors as there are data points.

As described, the overall problem solving strategy implemented by this QSPR planning agent represents an example of a filter method, i.e. one where filtering is done independently of the machine learning method to be applied afterwards. An alternative wrapper strategy, where the descriptor filtering is done using the same machine learning method, could be implemented as an alternative, competing QSPR planning agent.

QSPR model building

Each of the responses from the feature selection phase is available and eventually forwarded to each of the model building agents. The Discovery Bus currently has seven model building agents capable of building QSPR models for both classification, and continuous data using both the R programming environment (www.r-project.org) and Matlab (www.mathworks.com). An eighth agent is a Perl wrapped version of the Generalised, Unbiased, Interaction Detection and Estimation (GUIDE) software [28]. The QSPR models that are built by the model building agents are all validated by leave 10 % out cross-validation (using the folds carried through from the stratification stage); this goes some way to detect over-training and helps to ensure that only the best model for a given descriptor set from each model building agent is selected for further validation with the external test set.

RLinear (MLR) is a multiple linear regression model building agent that has the ability to create two MLR models. The first is obtained by using all of the descriptor inputs. A second model can also be produced if the number of descriptor inputs is less than 21; this model is obtained by backward stepwise selection of descriptors. Each new step (i.e. descriptor selection) involves building an additional MLR model in order to test whether it is better than any of the previously built models, thus performing this stepwise procedure with more than 21 descriptors becomes time consuming.

RNeuralNet (NN1) is based upon the package nnet of the R programming environment. The nnet algorithm uses a decay parameter to resist over-training and is parameterised by the number of hidden nodes required [29]. Our agent runs the algorithm many times with different weight parameters and different numbers of hidden nodes. The model with the combination of these parameters, which best explains the input data (by 10-fold cross-validation) is then selected.

RPLS (PLS) is a partial least squares algorithm based on the pls package of the R programming environment. This method involves fitting a number of latent variables to the response data. As well as the suggested model, our algorithm assesses a number of models around the suggested number of latent variables (models with one less than the suggested number, the suggested number, one more than the suggested number, up to four more than the suggested number).

Rrpart (CT) is based on the rpart package of the R programming language which builds classification and regression trees [30] by creating multiple models with differing complexity parameters and different numbers of observations at each of the terminal nodes. The rpart algorithm can be parameterised by constraints on node splitting size and leaf size. Our agent tries different values for the parameters, building many different trees and chooses those which best explain the data whilst minimising over-fitting.

Three agents are implemented using the Matlab programming language and described below.

GARMLR (GA1) implements an iteratively re-weighted least squares (IRLS) algorithm for robust multiple linear regression, wrapped by a genetic algorithm (GA) for feature selection. The GA system utilises chromosomes which are simply binary strings; each bit corresponding to one of the features under consideration. The 'fitness' of each individual (representing a selection of features) in the GA population is evaluated via a ten-fold cross validation of the robust multilinear modelling method with respect to the training set. The population evolves via the standard GA operations of selection, crossover and mutation [31, 32].

The GA uses a default population size of 100, making it unsuitable for modelling problems with very large numbers of features. The final model produced is a consensus of the fittest individuals from the final GA population.

GAWMLR (GA2) implements a weighted multiple linear regression, wrapped by the same GA algorithm as the GARMLR. The key difference between the two is that the GAWMLR algorithm is less resistant to the influence of outliers, as the 'robust' GARMLR is able to weight badly predicted points more effectively.

NetlabNN (NN2) is a consensus neural network model that implements an ensemble of multilayer perceptrons. Each perceptron has a single hidden layer, with the number of nodes allocated dynamically, depending on the numbers of training patterns and input features. Network, learning is performed by Bayesian regularisation, with the final ensemble output generated by a weighted mean of the individual perceptron outputs.

The final model agent is the GUIDE (MLS) multi-purpose machine learning algorithm designed and maintained at the University of Wisconsin-Madison [28]. GUIDE builds piecewise constant, best simple polynomial, multiple linear and stepwise regression models with univariate splits.

The QSPR models from each of the model building agents with the lowest relative mean square error values (Rel.MSE) are then cross-validated, after which testing with an external test set of data takes place to assess the true predictive quality of the model. The datasets have been previously separated into training-validation sets with an external test set by a stratification agent.

In addition to these classes of agents there are several other classes which take care of various minor aspects of the QSPR model discovery process, such as; formatting the input data to enable cross-validation to occur, selecting a test set of data from the input data to allow assessment of the true predictive nature of the resulting QSPR models, and producing graphical representations of the model statistics.

Results

To test the performance of the Discovery Bus in the automated discovery of QSPR models, three ADME properties, aqueous solubility [33], binding to human serum albumin [34] and the prediction of p-glycoprotein substrates [35], have been studied. These are previously published datasets which allow comparison of the performance of the Discovery Bus with that of human experts.

Several common statistical measures are used in the analysis of the quality of the resulting models; the most reliable statistic employed being the Rel.MSE values for cross-validation and test. A good model should have low and comparable Rel.MSE values for cross-validation and test, whilst having r^2 values as near to one as possible.

Aqueous solubility

Aqueous solubility is an important factor in determining the oral activity of a drug; as a consequence compounds with a low aqueous solubility tend to exhibit poor activity, even if

Table 1. Training and test statistics for aqueous solubility models.

Learner[1]	Filter[2]	Reduction		Types[3]	Linear Fit[4]	Training (1167)		Test (130)	
		Filter[5]	Learner[6]			Rel.MSE[7]	$r^{2,8}$	Rel.MSE[9]	$r^{2,10}$
MLS	H	1990 → 558	→ 54	R,D	1.46	0.11	0.89	0.12	0.89
	H	170 → 26	→ 14	A,E,H,D	0.13	0.11	0.89	0.13	0.88
	H	80 → 16	→ 12	A,H,D	0.14	0.11	0.88	0.12	0.87
	C	250 → 2	→ 2	A,R	0.18	0.13	0.87	0.16	0.84
	C	8 → 2	→ 2	A,L	0.16	0.13	0.87	0.16	0.86
GA1	H	80 → 16	→ 16	A,H,D	0.14	0.14	0.86	0.18	0.83
	C	8 → 2	→ 2	A,L	0.16	0.17	0.84	0.17	0.83
GA2	H	80 → 16	→ 16	A,H,D	0.14	0.14	0.79	0.17	0.83
	C	8 → 2	→ 2	A,L	0.16	0.17	0.84	0.17	0.83
NN1	H	250 → 54	→ 54	A,R	0.12	0.09	0.91	0.08	0.92
	H	80 → 16	→ 16	A,H,D	0.14	0.10	0.90	0.12	0.88
	H	326 → 46	→ 46	H,R,D	0.18	0.10	0.90	0.12	0.89
NN2	H	250 → 54	→ 54	A,R	0.12	0.08	0.92	0.09	0.91
	H	326 → 46	→ 46	H,R,D	0.18	0.10	0.90	0.11	0.90
	H	86 → 16	→ 16	A,H,D	0.14	0.11	0.89	0.11	0.89
MLR	H	81 → 58	→ 58	A,H,R,D	0.12	0.12	0.88	0.14	0.87
	H	80 → 16	→ 11	A,H,D	0.14	0.14	0.84	0.17	0.84
	C	8 → 2	→ 2	A,L	0.16	0.16	0.85	0.18	0.83
PLS	Hr10	1992 → 112	→ (7)	A,R,L,D	0.13	0.12	0.88	0.14	0.87
	H	170 → 26	→ (8)	A,E,H,D	0.13	0.13	0.87	0.15	0.86
	H	80 → 16	→ (8)	A,H,D	0.14	0.14	0.86	0.29	0.79
CT	C−1	80 → 1	→ 1	A	0.28	0.37	0.63	0.39	0.66
	H	8 → 2	→ 2	A,L	0.16	0.43	0.57	0.27	0.77

1 The model building agent.
2 The filter used to produce the training feed into the learner agent.
3 The types of descriptors used in the final model. A – ACD, D – Dragon, E- E-State indices, H - HE-state indices, R – RECON, L–LSER.
4 The measure of the goodness of fit of a linear model built from the same set of descriptors as were passed to the learner. It is the cross validation relative mean squared error for the linear model built from the same stratified training feed.
5 Reduction in the number of descriptors achieved by the filter. In the entries for PLS the figures in parentheses give the number of latent variables used.
6 Reduction in the number of descriptors achieved by the learner.
7 Relative mean square error from the 10-fold cross validation of the model.
8 The r^2 for the 10-fold cross validation of the model.
9 The relative mean squared error when the model is tested on the independent test set.
10 The r^2 when the model is tested on the independent test set.

they have high intrinsic activity. The dataset chosen to study this property consisted of 1297 diverse organic compounds compiled by Huuskonen. In this study 129 of the compounds were selected as a test set, leaving 1168 for model building.

Seven of the eight model building techniques performed well (Table 1). The two best QSPR models being constructed by the neural network agents, the regression tree agent performed significantly worse, constructing a less predictive QSPR model than the other model building agents.

The QSPR models generated by the Discovery Bus are comparable with those produced in the original study by Huuskonen [36], in which two QSPR models were generated, one using a neural network ($r^2_{cv} = 0.94$, $r^2_{test} = 0.92$) and a second using multiple linear regression ($r^2_{cv} = 0.89$, $r^2_{test} = 0.88$). Figure 3 shows predicted versus actual values for one of the NN1 QSPR models generated by the Discovery Bus.

Binding to Human Serum Albumin (HSA)

HSA binding is important in the transport of drugs around the body and is factor in a drug's ability to diffuse from the circulatory system to its target tissues [37]. A data set of 91 diverse drug, and drug-like compounds were used to construct QSPR models [34]. The best models were produced by the NN1 and NN2 agents, with the MLS and CT agents performing poorly, Table 2. The best two models compare favourably with those previously published (literature values: $r^2_{cv} = 0.78$, $r^2_{test} = 0.73$) [34] with the best model shown in Figure 4

P-glycoprotein

P-glycoprotein (P-gp) is an ATP-dependent efflux pump capable of transporting a wide range of chemicals across the plasma membrane in cells [38, 39]. Over-expression can lead to drug resistance, causing the efficacy of chemotherapy [40, 41] as well as antiviral and antibiotic therapies [42] to be

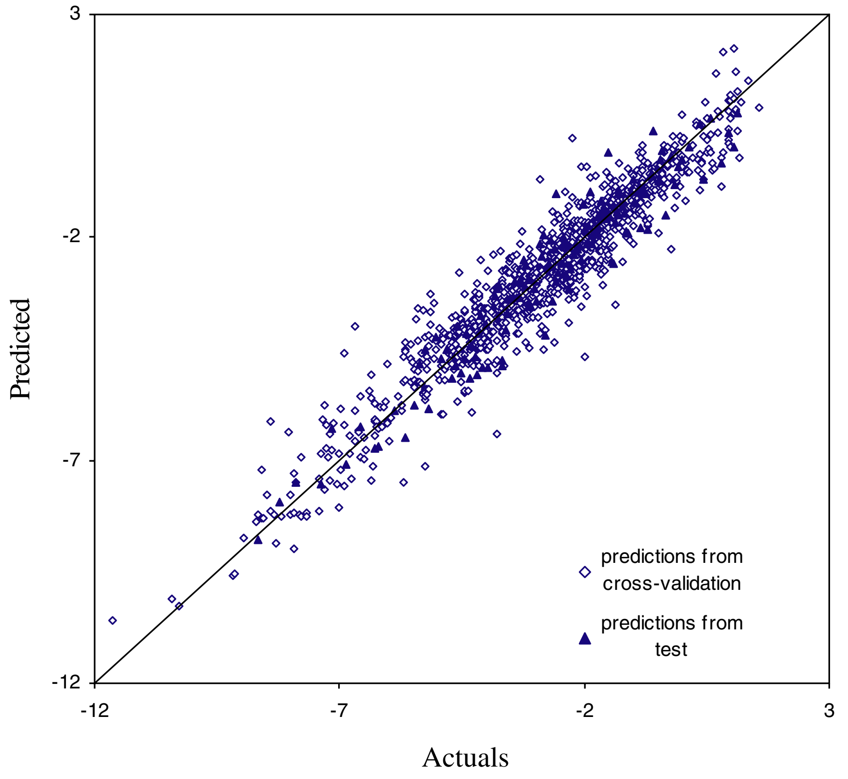

Figure 3. Predicted versus Actual Solubility Values for NN1 Model.

Table 2. Training and test statistics for binding to human serum albumin model (for explanation see footnotes to Table 1).

Learner	Filter	Reduction		Types	Linear Fit	Training (82)		Test (9)	
		Filter	Learner			Rel.MSE	r^2	Rel.MSE	r^2
MLS	Hh2	332 → 39	→ 8	A,E,R	0.92	0.40	0.62	0.25	0.81
	H	250 → 59	→ 12	A,R	1.62	0.47	0.56	0.30	0.76
	Hh4	382 → 20	→ 1	A	0.25	0.50	0.50	0.57	0.49
GA1	Hh2	1998 → 39	→ 26	A,R,D	0.42	0.23	0.77	0.20	0.85
	Hh4	344 → 20	→ 19	H,R,D	0.42	0.26	0.74	0.28	0.78
	Hh10	302 → 9	→ 9	H,R	0.27	0.27	0.73	0.40	0.64
GA2	Hh4	1998 → 20	→ 19	A,R,D	0.37	0.25	0.75	0.30	0.70
	H	302 → 19	→ 17	H,R	0.52	0.27	0.73	0.40	0.66
	Hh10	302 → 9	→ 9	H,R	0.27	0.28	0.72	0.42	0.60
NN1	H	8 → 5	→ 5	A,L	0.37	0.17	0.83	0.15	0.87
	Hh10	346 → 8	→ 8	A,R,D	0.30	0.30	0.70	0.16	0.84
	H	302 → 19	→ 19	H,R	0.27	0.32	0.70	0.39	0.71
NN2	H	8 → 5	→ 5	A,L	0.37	0.19	0.81	0.12	0.88
	Hh2	1998 → 39	→ 39	A,R,D	0.42	0.24	0.76	0.16	0.90
	Hh10	302 → 9	→ 9	H,R	0.26	0.26	0.74	0.35	0.65
MLR	Hh4	1998 → 20	→ 13	A,R,D	0.23	0.23	0.77	0.26	0.74
	Hh4	346 → 20	→ 8	A,H,R,D	0.25	0.25	0.75	0.29	0.71
	Hh4	344 → 20	→ 9	H, R, D	0.25	0.25	0.75	0.30	0.77
	Hh4	382 → 20	→ 11	A,H,R,L	0.25	0.25	0.75	0.26	0.77
	Hh4	332 → 20	→ 10	A,R,L	0.26	0.26	0.75	0.35	0.76
PLS	Hh4	1998 → 20	→ (4)	A,R,D	0.33	0.24	0.76	0.38	0.63
	Hh10	1998 → 8	→ (4)	A,R,D	0.32	0.29	0.71	0.36	0.69
	Hh2	1998 → 39	→ (6)	A,R,D	0.42	0.24	0.76	0.25	0.75
CT	C	78 → 2	→ 2	A	0.45	0.71	0.32	0.28	0.74
	C	302 → 2	→ 2	R	0.41	0.71	0.30	0.30	0.71
	C	172 → 2	→ 2	D	0.47	0.72	0.33	1.49	0.10

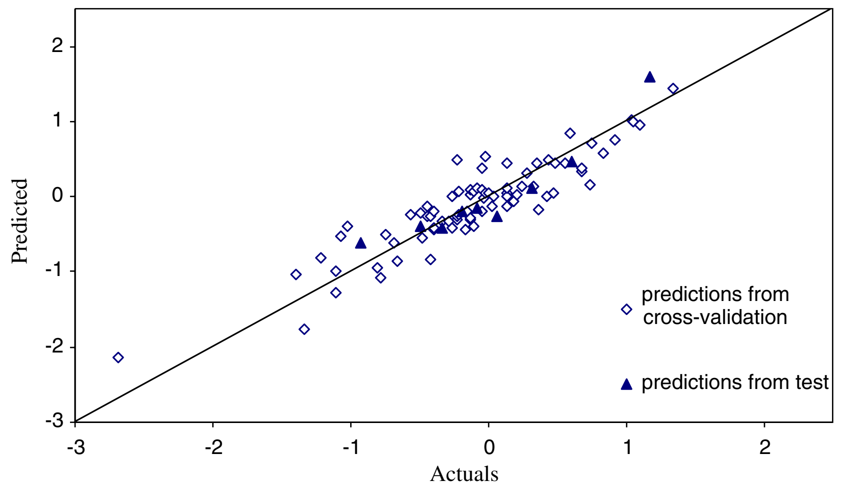

Figure 4. Predicted versus Actual Human Serum Albumen binding values for the GA2 Model.

Table 3. Overall Performance of Neural Net (NN2) and Classification (CT) Methods for P-gp Substrates.

Technique	% Correctly classified training set	% Correctly classified test set
NN2	95.6	69.7
CT	90.4	81.0

reduced. A data set comprising 184 compounds used in a previous study [35] was used to build classification QSPR models. Only two of the agents are capable of building this type of model; NN2 and CT. The best results of which are shown in Table 3.

Both of these models are comparable with a support vector machine classification model [35], in which 80% of the compounds in the test set were correctly classified. It is interesting to note that the CT agents' best model is better than previously reported classification models using k-nearest neighbour [43] (70.8%), probabilistic neural network (74.4%), and a C4.5 decision tree [31] (71.5%) methods. However, the ability of the model to classify the inactive compounds is worse (50% correctly classified). This is likely to be due to the difference in the training data used in the present study, since 16 fewer compounds were available from an in-house database of drug like structures, all of which belong to the inactive class. It is likely therefore that if more inactive compounds were added to the training data then the percentage of correctly classified inactive compounds would be significantly higher. (Tables 4 and 5).

Discussion

The results show that an automated QSPR modelling process implemented on the Discovery Bus is able to reproduce previously published QSPR models for a set of three important ADME properties. The QSPR model building and testing process described above is fully automated and requires no user decisions about model building technique, descriptor selection or filtering methods.

Descriptor space

The competitive workflow environment of the Discovery Bus allows thorough exploration of the available descriptor space during the QSPR model discovery process. This is because it is possible to add unlimited descriptor agents to the molecular descriptor class, whilst leaving the selection of the best set of molecular descriptors for a given property to the agents in the filter features and machine learning classes. A good example of the power of this approach to find the best possible set of descriptors is in the classification model developed for P-gp. The best model built by the CT agent is able to classify active compounds better than a number of other well established classification methods. One possible explanation is that the ability to thoroughly explore the available descriptor space by building a range of models based on differing sets of descriptors is responsible for the ability of the CT agent to produce a good classification model.

The addition of new and improved agents to the molecular descriptors class enables the continuous improvement of the available descriptor space, which is expected to improve the ability of the agents within the model building class to produce better QSPR models.

Model space

Having multiple agents within the molecular descriptor class enables exploration of the

Table 4. Performance of Neural Net (NN2) Method for Classification of P-gp Substrates.

Classification	Train		Test	
	Active	Inactive	Active	Inactive
Active	56	0	7	3
Inactive	3	9	3	5
True Positive/True Negative Ratio	Train		Test	
	Correct	Incorrect	Correct	Incorrect
Active	1.00	0.00	0.70	0.30
Inactive	0.75	0.25	0.62	0.38

Table 5. Performance of Classification Tree (CT) Method for P-gp Substrates.

Classification	Train		Test	
	Active	Inactive	Active	Inactive
Active	56	4	9	1
Inactive	3	10	4	4
True Positive/True Negative Ratio	Train		Test	
	Correctly class	Incorrect class	Correctly class	Incorrect class
Active	0.93	0.07	0.90	0.10
Inactive	0.77	0.23	0.50	0.50

descriptor space and this in turn allows the multiple machine learning agents to fully explore the available model space. During the course of the QSPR model discovery process, in excess of a hundred models are typically built, cross-validated and tested against an external data set (many more models are built but are of insufficient quality for cross-validation and testing). Exploration of the model space allows differing combinations of descriptors with machine learning techniques to be found which are predictive in differing areas of the model and hence, chemical space.

It is noteworthy that exploration of the model space can lead to new opportunities to discover local QSPR models within a global model. By constructing many models across model space, predicted values of test compounds are produced by all of the best models across the model space; ultimately selecting the model which best predicts each test compound. If the test set of compounds were diverse, it is likely that different compounds will use models from different areas of the model space Alternatively, groups of models that describe similar areas of models space could be selected [44]

Feature selection

There are two principal strategies for variable reduction: filter and wrapper methods [27]. In the competitive workflow architecture of the Discovery Bus, both strategies can be employed in competition with one another. In the examples shown preliminary results suggest that a combination of the two filtering techniques appears to lead to better QSPR models. Initial filtering methods remove correlated and redundant molecular descriptors, producing a number of differently filtered descriptor sets. These sets are then used as input to traditional QSPR model building techniques such as MLR and PLS as well as machine learning wrapped feature selection methods such as NN and GA.

It is also possible to select only a specific set of molecular descriptors (from more than a single agent if required) for model building. This additional functionality also allows for more traditional QSPR model building, should the modeller wish to retain a degree of control over the descriptor space explored.

An interesting question for future research is to what extent the current implementation, with relatively few agents and datasets, scale to conditions where there is a much wider range of data as well as significantly more agents available. Although additional CPU resource could be assigned, the combinatorial nature of the Competitive Workflow architecture suggests that conditions where agents are forced to compete for resource should be expected. In those conditions, we expect the focus of research to shift from the detection of specific QSAR models to the issue of learning in multi-agent environments [9]. In effect, in this early stage of development, since all agents can run to completion, the QSAR modelling workflow is implemented as a "brute-force search". However, this will not be a suitable strategy as the Discovery Bus becomes more loaded with agents, whatever the CPU resource available. As a first step towards understanding this issue, we have implemented a learning reinforcement feedback mechanism in the Discovery Bus kernel, but have not yet had an opportunity to explore its potential.

Conclusions

An autonomous QSPR modelling system has been described which is able to reproduce the quality of models generated in earlier studies, without any manual direction or intervention, expert or otherwise. The process is completed within hours, and exhaustively explores the available model and descriptor space. The process is easy to update and expand, through the addition of new data, algorithms and strategies, and generates ensembles of QSPR models from global input data. No pre-judgments are required as to which descriptor types or learning methods to deploy, and the best models are automatically selected. New algorithms and data are automatically evaluated against historical results and automated predictions of properties for novel compounds by ensembles of models are a feature of the system.

The competitive workflow architecture creates an environment for the assessment of competing QSPR modelling technologies and facilitates the exploration of model space. It models the problem solving process of the human QSPR expert, and models the acquisition of human experience by the implementation of reinforcement mechanisms for successful modelling pathways in a competitive workflow. Although we have only demonstrated modelling success for three ADME properties, there is no obvious reason why the process would not be equally successful with other ADME, toxicity or biological properties and application to a more comprehensive range of property types is currently underway.

The Discovery Bus automates the exchange of requests for information and responses at the strategy as well as the sub-task agent level. It has a generality which will enable it to drive other complementary problem solving strategies such as ones for Lead Identification or Lead Generation.

In this way, the Discovery Bus provides a mechanism for the automation of a wide range of molecular design workflows, freeing the human expert to concentrate on the design and implementation of novel algorithms, rather than the routine processing of information.

Note

1 Absorption, Distribution, Metabolism and Elimination.

Acknowledgements

We thank Dr Simon Thomas and Dr Graham Searle for implementation of the Matlab Neural Network and Genetic Algorithm-wrapped agents respectively.

References

1. van der Aalst, W.M.P., Barros, A.P., ter Hofstede, A.H.M. and Kiepuszewski, B., Advanced workflow, In Etzion, O. and Scheuermann, P. (Eds.), 7th International Conference on Cooperative Information Systems, 2000. Springer-Verlag, Berlin, 2000, pp. 18–29.
2. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B. and Barros, A.P., Distributed Parallel Databases, 14 (2003) 5.
3. Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., Greenwood, M., Carver, T., Glover, K., Pocock, M.R., Wipat, A. and Li, P., Bioinformatics, 20 (2004) 3045.
4. Oinn, T. M., Addis, M., Ferris, J., Marvin, D., Greenwood, T., Carver, T., Wipat, A. and Li, P. In Taverna, Lessons in Creating a Workflow Environment for the Life Sciences, 10th Global Grid Forum, Berlin, 2004.
5. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludaescher, B. and Mock, S. In Kepler: Towards a Grid-enabled System for Scientific Workflows, 10th Global Grid Forum, Berlin, 2004.
6. Corkill, D.D. In Collaborating Software: Blackboard and Multi-agent Systems & The Future. Proceedings of the National Lisp Conference, New York, 2003.
7. Repetto, M., Paolucci, M. and Boccalatte, A. In A Design Tool to Develop Agent-based Workflow Management Systems, 4th AI*IA/TABOO Joint Workshop "From Objects to Agents": Intelligent Systems and Pervasive Computing, Villasimius, Italy, 2003; Pitagora Editrice Bologna.
8. Ramampiaro, H., Wang, A. I. and Brasethvik, T. In Supporting distributed cooperative work in cagis, 7th European Workshop Software Process Technology, Kaprun, Austria, 2000; Conradi, R. Springer: Kaprun, Austria, p. 115.
9. Weiss, G. and Sen, S., Learning in multiagent systems, In Weiss, G. (Ed), Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence260. MIT Press, Cambridge, MA, 1999.
10. Corradini, F., Mariani, L. and Merelli, E., Int. J. Software Tools Technol. Transfer, 6 (2004) 231.
11. Armano, G., Milanesi, L. and Orro, A., In An agent architecture for predicting protein secondary structure, NETTAB 2002: Agents in Bioinformatics, Bologna, Italy, 2002.

12. Peleg, M.,Yeh, I. and Altman, R., Bioinformatics, 18 (2002) 825.
13. Olah, M., Bologa, C. and Oprea, T.I., J. Comput. Aided Mol. Des., 18 (2004) 437.
14. Kovatcheva, A., Golbraikh, A., Oloff, S., Xiao, Y. D., Zheng, W., Wolschann, P., Buchbauer, G. and Tropsha, A., J. Chem. Inf. Comput. Sci., 44 (2004) 582.
15. Ferber, J. Multi-agent Systems. An Introduction to Distributed Artificial Intelligence. Pearson Education Limited, London, 1999.
16. Tarbit, M.H. and Berman, J., Curr. Opin. Chem. Biol., 2 (1998) 411.
17. Rodrigues, A.D., Biochem. Pharmacol., 48 (1994) 2147.
18. Eddershaw, P.J. and Dickins, M., Pharm. Sci. Technol. Today, 2 (1999) 13.
19. van de Waterbeemd, H., Curr. Opin. Drug Discov. Devel, 5 (2002) 33.
20. Leach, A.R. and Gillet, V.J. An introduction to chemoinformatics. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 77–101.
21. ACD Physchem batch, http://www.acdlabs.com/products/
22. Todeschini, R., Consonni, V., Mauri A. and Pavan M., 2004. Dragon plus version 5.1.
23. Hall, L.H. and Kier, L.B., J. Chem. Inf. Comput. Sci., 35 (1995) 1039.
24. Abraham, M.H., Chem. Soc. Rev., 22 (1993) 73.
25. Breneman, C.M., Sundling, C.M., Sukumar, N., Lingling, S., Katt, W.P. and Embrechts, M.J., J. Comput. Aided Mol. Design, 17 (2003) 231.
26. Hall, M.A. In Correlation-based feature selection of discrete and numeric class machine learning, Proceedings of the International Conference on Machine Learning, San Francisco, CA, 2000. Morgan Kaufmann, San Francisco, CA, 2000, pp. 359–366.
27. Hall, M.A. Correlation-based Feature Selection for Machine Learning. Waikato, New Zealand, 1999.
28. Loh, W.-Y., Statistica Sinica, 12 (2002) 361–386.
29. Venables, W.N. and Ripley, B., Modern applied statistics with S. 4th ed.; Springer: 2002.
30. Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J., Classification and regression trees. ed.; 1984.
31. Yang, J. and Honavar, V., IEEE Trans. Intelligent Syst., 13 (1998) 44.
32. So, S.S. and Karplus, M., J. Med. Chem., 39 (1996) 152.
33. Huuskonen, J., J. Chem. Inf. Comput. Sci., 40 (2000) 733.
34. Colmenarejo, G., Alvarez-Pedraglio, A. and Lavandera, J., J. Med. Chem., 44 (2001) 4370.
35. Xue, Y., Yap, C.W., Sun, L.Z., Cao, Z.W., Wang, J.F. and Chen, Y.Z., J. Chem. Inf. Comput. Sci., 44 (2004) 1497.
36. Huuskonen, J.J., Livingstone, D.J. and Tetko, I.V., J. Chem. Inf. Comput. Sci., 40 (2000) 947.
37. Herve, F., Urien, S., Albengres, E., Duche, J.-C. and Tillement, J., Clin. PharmacoKinet., 26 (1994) 44.
38. Schmitt, L. and Tampe, R., Curr. Opin. Struct. Biol., 12 (2002) 754.
39. van Veen, H.W. and Konings, W.N., Adv. Exp. Med. Biol., 456 (1998) 145.
40. Gottesman, M.M., Pastan, I. and Ambudker, S.V., Curr. Opin. Genet. Dev., 6 (1996) 610.
41. Ambudkar, S.V., Dey, S., Hrycyna, C.A., Ramachandra, M., Pastan, I. and Gottesman, M.M., Annu. Rev. Pharmacol. Toxicol., 39 (1999) 361.
42. Delph, Y. P-glycoprotein: A tangled web waiting to be unraveled. http://www.aidsinfonyc.org/tag/science/pgp.html.
43. Huberty, C. J., Applied discriminant analysis. ed.; John Wiley & Sons: New York, 1994.
44. Izrailev, S. and Agrafiotis, D.K., J. Mol. Graph. Mod., 22 (2004) 275.