Reoptimization of MDL Keys for Use in Drug Discovery

Joseph L. Durant,* Burton A. Leland, Douglas R. Henry, and James G. Nourse

MDL Information Systems, 14600 Catalina Street, San Leandro, California 94577

Received December 17, 2001

For a number of years MDL products have exposed both 166 bit and 960 bit keysets based on 2D descriptors. These keysets were originally constructed and optimized for substructure searching. We report on improvements in the performance of MDL keysets which are reoptimized for use in molecular similarity. Classification performance for a test data set of 957 compounds was increased from 0.65 for the 166 bit keyset and 0.67 for the 960 bit keyset to 0.71 for a surprisal S/N pruned keyset containing 208 bits and 0.71 for a genetic algorithm optimized keyset containing 548 bits. We present an overview of the underlying technology supporting the definition of descriptors and the encoding of these descriptors into keysets. This technology allows definition of descriptors as combinations of atom properties, bond properties, and atomic neighborhoods at various topological separations as well as supporting a number of custom descriptors. These descriptors can then be used to set one or more bits in a keyset. We constructed various keysets and optimized their performance in clustering bioactive substances. Performance was measured using methodology developed by Briem and Lessel. "Directed pruning" was carried out by eliminating bits from the keysets on the basis of random selection, values of the surprisal of the bit, or values of the surprisal S/N ratio of the bit. The random pruning experiment highlighted the insensitivity of keyset performance for keyset lengths of more than 1000 bits. Contrary to initial expectations, pruning on the basis of the surprisal values of the various bits resulted in keysets which underperformed those resulting from random pruning. In contrast, pruning on the basis of the surprisal S/N ratio was found to yield keysets which performed better than those resulting from random pruning. We also explored the use of genetic algorithms in the selection of optimal keysets. Once more the performance was only a weak function of keyset size, and the optimizations failed to identify a single globally optimal keyset. Instead multiple, equally optimal keysets could be produced which had relatively low overlap of the descriptors they encoded.

INTRODUCTION

There are a number of tasks in the drug-discovery workflow which require grouping and/or separating molecules based on some criteria. Clustering[1,2] and diversity analysis[3,4] are often implemented using distances or similarity measures based on binary keysets. Databases and libraries can be characterized based on statistical distribution functions[5] of bits within keysets, in addition to the more familiar calculations based on Pfizer "Rule of Five",[6] distribution functions,[7,8] or atom occurrence counts.[9] Lead identification using fingerprints based on binary keysets continues to be examined.[10−12] Finally, the related topics of library design[13,14] and optimal compound selection[15−17] often use binary keysets.

In this paper we will use the following definitions to refer to keyset components. "Descriptor" will be used to refer to a molecular feature. Descriptors can be encoded into binary "keybits". There can be a one-to-one relationship between descriptors and keybits, or hashing can be used to create a many-to-one or many-to-many relationship between descriptors and keybits. An ordered collection of keybits constitute a "keyset". "Keys" is context-sensitive and is used to refer either to keybits or keysets.

Historically, molecular keysets have been used for substructure searching.[18,19] For example, a number of topological features of a query molecule can be used to set keybits; a search for molecules matching that query could then be facilitated by screening out all molecules in the database which do not set those same keybits. An optimization strategy for such a task might include the choice of properties to encode which would be structure differentiating and arrangement of them within the keyset in order of increasing probability of occurrence within a reference data set.

Indeed, this is the genesis of two common MDL keysets: one containing 960 keybits and the other containing a subset of 166 keybits.[20] Their use in substructure searching has been largely obviated by index-based search techniques.[21] They have, however, found a variety of uses in the drug-discovery workflow.

There continues to be interest in the design of keysets which will provide more performance in a drug-discovery workflow. Recent work includes examination of *in vitro* affinity fingerprints,[22,23] *in silico* (or virtual) affinity fingerprints,[24−26] and feature trees[27] as methods of producing keysets optimized for similarity searches. Briem and Lessel[26] carried out a study comparing various 2D descriptors with several virtual affinity fingerprints. They found that the 2D descriptors out-performed the affinity fingerprint methods.

The existing MDL 2D keyset technology can, in theory, produce in excess of 3 million distinct keybits, which can be combined into innumerable keysets. We wanted to explore the reoptimization of keysets and determine how much performance could be improved by such reoptimization.

* Corresponding author phone: (510)895-1313; e-mail: jdurant@mdli.com.

10.1021/ci010132r CCC: $22.00
Published on Web 09/14/2002

Table 1. Atom-Based Properties

n	P
0	null
1	atom with at least three neighbors
2	heteroatom
3	atom involved in one or more multiple bonds, not aromatic
4	atom with at least four neighbors
5	atom with at least two heteroatom neighbors
6	atom with at least three heteroatom neighbors
7	heteroatom with at least one hydrogen attached
8	carbon with at least two single bonds and at least two hydrogens attached
9	carbon with at least one single bond and at least three hydrogens attached
10	halogen
11	atom has at least three single bonds
12	atom is in at least two different six-membered rings
13	not used
14	atom has more than two ring bonds
15	atom is at a ring/chain boundary. When a comparison is done with another atom the path passes through the chain bond.
16	atom is at an aromatic/nonaromatic boundary. When a comparison is done with another atom the path passes through the aromatic bond.
17	atom with more than one chain bond
18	atom is at a ring/chain boundary. When a comparison is done with another atom the path passes through the ring bond.
19	atom is at an aromatic/nonaromatic boundary. When a comparison is done with another atom the path passes through the nonaromatic bond.
20	atom is a heteroatom in a ring.
21	rare properties: atom with five or more neighbors, atom in four or more rings, or atom types other than H, C, N, O, S, F, Cl, Br, or I
22	rare properties: atom has a charge, is an isotope, has two or more multiple bonds, or has a triple bond.
23	nitrogen
24	sulfur
25	oxygen
26	not used
27	not used
28	not used
29	not used
30	atom has two neighbors, each with three or more neighbors (including the central atom).
31	atom has two hydrocarbon (CH_2) neighbors

Table 2. Single-Atom Atom-Based Properties

n	A
0	null
1	atom with at least three neighbors
2	heteroatom
3	atom involved in some multiple bonds, not aromatic
4	atom with at least four neighbors
5	atom with at least two heteroatom neighbors
6	atom with at least three heteroatom neighbors
7	heteroatom with at least one hydrogen attached
8	carbon with at least two single bonds and at least two hydrogens attached
9	carbon atom in a C═C double bond
10	atom has at least two single bonds
11	atom has at least three single bonds
12	atom is in at least two different six-membered rings
13	not used
14	atom has more than two ring bonds
15	atom is at a ring/chain boundary
16	central atom is at an aromatic/nonaromatic boundary
17	atom with more than one chain bond
18	atom is in a ring
19	aromatic atom
20	atom is a heteroatom in a ring.
21	rare properties: atom with five or more neighbors, atom in four or more rings, or atom types other than H, C, N, O, S, F, Cl, Br, or I
22	rare properties: atom has a charge, is an isotope, has two or more multiple bonds, or has a triple bond.
23	nitrogen
24	sulfur
25	oxygen
26	atom in a three-membered ring
27	atom in a four-membered ring
28	atom in a five-membered ring
29	atom in a six-membered ring
30	atom has two neighbors, each with three or more neighbors (including the central atom).
31	atom has two hydrocarbon (CH_2) neighbors

COMPUTATIONAL METHOD

Structure of MDL Keys. The underlying technology used in the MDL 2D keysets is based on a general molecule perception algorithm, which perceives a number of atom, bond, and custom properties. The mapping of these properties into descriptors and then into the keybits and ultimately keysets is under software control.

Specifically, a keybit is defined by nine numbers, which we will denote by n1 through n9. The first four numbers encode the various properties into descriptors. The remaining five numbers determine which keybits are set by the descriptor.

General Definitions. MDL has specific definitions for a number of general chemistry terms. "Heteroatom" is used to refer to any non-C, non-H atom, and is abbreviated "Q". "Halogens" consist of F, Cl, Br, and I. "Other" atoms include any atoms other than H, C, N, O, Si, P, S, F, Cl, Br, and I and is abbreviated "X". "Aromatic" refers to bonds which are either "Kekule aromatic" or "arom5". "Kekule aromatic" bonds are those in a six-membered ring system with alternating double and single bonds or the perimeter bonds of an azulene. "Arom5" bonds are those in a five-membered ring with two double bonds and a "heteroatom" or C^- at the apex of the ring.

Descriptor Encoding. The largest block of descriptors makes use of algorithmically calculated atom-based properties. Specifically we perceive 26 properties of type P, as listed in Table 1, and 30 properties of type A (most of which are the same as type P properties), as listed in Table 2. Additionally we perceive 32 one-atom environments, as listed in Table 3.

One class of possible atom-based descriptors result from the presence of one or two properties of type A located on a single atom. For this class n1 = 0; n2 and n3 encode the appropriate values of A.

A second class of possible atom-based descriptors result from the presence of an atom with property P, separated by one to four bonds from a second atom with a property P′. For this class n1 encodes the number of bonds between the atoms, and n2 and n3 encode the appropriate values of P and P′.

A third class of possible atom-based descriptors results from the presence of an atom with property A located in the center of a particular atom environment. In this case n1 = 7, n2 encodes the atom environment, and n3 encodes the property A.

Another block of descriptors encodes one of 264 atom-bond-atom combinations. Atoms include C, N, O, Si, P, S,

Table 3. Atomic Environments

n	atom environment[a]	n	atom environment[a]
0	C(CC)	16	Q(CC)
1	C(CCC)	17	Q(CCC)
2	C(CN)	18	Q(CN)
3	C(CCN)	19	Q(CCN)
4	C(NN)	20	Q(NN)
5	C(NNC)	21	Q(CNN)
6	C(NNN)	22	Q(NNN)
7	C(CO)	23	Q(CO)
8	C(CCO)	24	Q(CCO)
9	C(NO)	25	Q(NO)
10	C(NCO)	26	Q(CNO)
11	C(NNO)	27	Q(NNO)
12	C(OO)	28	Q(OO)
13	C(COO)	29	Q(COO)
14	C(NOO)	30	Q(NOO)
15	C(OOO)	31	Q(OOO)

[a] The first symbol is the central atom, with atoms bonded to the central atom listed in parentheses. "Q" is any non-C, non-H atom. If only two atoms are in parentheses, there is no implication concerning the other atoms bonded to the central atom.

F, Cl, Br, I, and "other" (X), with bond types of "single" (−), "double" (=), "triple" (#), and "ring" (%). These descriptors have n1 = 6; n2 and n3 are set as shown in Table 4.

Yet another block encodes a number of custom and Sgroup features.[28] These can be found in Table 5.

The first subblock of these descriptors encodes a series of properties which are used in the 166 bit MDL keyset. The next subblock of 256 descriptors encodes atom types of 1−256. Normally atom types between 1 and 103 correspond directly with periodic table elements. The range 104−256 allows encoding of custom atom types.

A final subblock of descriptors encode a variety of MDL Sgroup properties.

In converting these descriptors to keybits we also need to set an occurrence count, which is encoded in n4. This allows us to define descriptors for occurrence counts of "1 or more" up to "999 or more".

KeyBit Encoding. The last five numbers in the keybit definition are used to control which keybit(s) are set by the descriptor defined by the first four numbers. N5 is used to specify the number of keybits which are set, which is in the range of 1 to 3. N6 is a flag indicating whether (1) or not (0) the keybit(s) set are also set by other descriptors. The final three numbers, n7, n8, and n9, identify the keybits, with "0" used for padding.

Putting it Together. At this point we can consider the meaning of an example keybit:

position:	n1	n2	n3	n4	n5	n6	n7	n8	n9
key:	2	3	5	2	3	1	479	469	763

This keybit can be translated to the following: At least two occurrences (n4) of an atom in a multiple, nonaromatic bond (n2) located two bonds (n1) away from an atom with at least two heteroatom neighbors (n3). The descriptor sets three keybits (n5), which can be set by other descriptors (n6). The keybits set are 479, 469, and 763 (n7−n9).

A number of properties are incompatible, resulting in the fact that fewer descriptors are chemically possible than are mathematically allowed. Additionally, a number of descriptors can be encoded two different ways, most frequently by reversing n2 and n3. The algorithm used will only set keybits for n2 less than or equal to n3 in these instances. As a result we can encode 3234 different descriptors encoding occurrence counts of "1 or more". Since these descriptors can have occurrence counts of "1 or more" to "999 or more", we have the ability, in theory, to produce in excess of 3 million distinct descriptors, which can be combined into innumerable keysets. Our question is then "How does one select an optimal set of descriptors for use in drug discovery workflows, or, more concretely, for use in molecular similarity calculations?".

Success Measure. We chose to adopt a success measure defined by Briem and Lessel.[26] Their success measure is calculated by taking the mean fraction of molecular nearest neighbors which are of the same activity class as the target active compound, and averaging this quantity over all active target compounds. This measure should be directly applicable to clustering and diversity applications and should also reflect performance expected in other drug discovery applications.

To implement this method Briem and Lessel have defined a set of six categories of molecules selected from the MDDR database.[29] These molecules consist of 134 PAF antagonists, 49 5-HT3 antagonists, 49 TXA2 antagonists, 40 ACE inhibitors, 111 HMG-CoA reductase inhibitors, and 574 other compounds selected at random from MDDR which were not primarily members of the already selected activity classes. The first five categories will be referred to as "actives", the randomly chosen molecules as "inactives". The total number of molecules in the data set is 957.

The next step in calculating the success measure for the keyset under investigation is to collect nearest neighbors for all the compounds in the "active" classes. In the original paper Euclidean distances between keys were used; we use the computationally easier Hamming distance, defined as the number of bits which are different between the two bit sets. For binary keys the Euclidean distance is the square root of the Hamming distance. Use of the Hamming distance results in the same nearest neighbors as those found using a Euclidean metric. The fraction of the top 10 nearest-neighbors which are in the same activity class as the target defines the success measure for that target molecule. The total success measure is then the average of the individual success measures for all the active molecules. We have added explicit consideration of ties by including all molecules which have distances equal to the tenth nearest neighbor. This removes a dependence on the order that the molecules are considered which was present in the original method. It also has the effect of reducing the success measure for the 166 keybit MDL keyset to 0.65 from the 0.67 reported by Briem and Lessel.[26]

Completely random compound selection of molecules will yield a nonzero fraction of compounds of the same activity class within the 10 nearest neighbors. Consideration of the average probability of this occurring for the five active classes yields an average success measure of 0.08 for random selection.

Use of Hamming distances in this context most probably results in a performance degradation relative to results using Tanimoto distances. However, calculation of Hamming distances can be easily optimized, and the resulting optimized code gave us increased latitude in carrying out GA studies (vide infra).

Table 4. Atom-Bond-Atom Properties

n2	n3		n2	n3		n2	n3		n2	n3		n2	n3	
0	17	C−C	4	8	Br−Br	11	6	P═P	18	4	S#S	25	2	N%N
0	18	C−N	4	9	Br−Si	11	7	P═F	18	5	S#Cl	25	3	N%O
0	19	C−O	4	10	Br−I	11	8	P═Br	18	6	S#P	25	4	N%S
0	20	C−S	4	15	Br-X	11	9	P═Si	18	7	S#F	25	5	N%Cl
0	21	C−Cl	4	25	Si−Si	11	10	P═I	18	8	S#Br	25	6	N%P
0	22	C−P	4	26	Si−I	11	15	P═X	18	9	S#Si	25	7	N%F
0	23	C−F	4	31	Si-X	11	23	F═F	18	10	S#I	25	8	N%Br
0	24	C−Br	5	10	I−I	11	24	F═Br	18	15	S#X	25	9	N%Si
0	25	C−Si	5	15	I−X	11	25	F═Si	18	21	Cl#Cl	25	10	N%I
0	26	C−I	7	31	X−X	11	26	F═I	18	22	Cl#P	25	15	N%X
0	31	C−X	8	17	C═C	11	31	F═X	18	23	Cl#F	25	19	O%O
1	2	N−N	8	18	C═N	12	8	Br═Br	18	24	Cl#Br	25	20	O%S
1	3	N−O	8	19	C═O	12	9	Br═Si	18	25	Cl#Si	25	21	O%Cl
1	4	N−S	8	20	C═S	12	10	Br═I	18	26	Cl#I	25	22	O%P
1	5	N−Cl	8	21	C═Cl	12	15	Br═X	18	31	Cl#X	25	23	O%F
1	6	N−P	8	22	C═P	12	25	Si═Si	19	6	P#P	25	24	O%Br
1	7	N−F	8	23	C═F	12	26	Si═I	19	7	P#F	25	25	O%Si
1	8	N−Br	8	24	C═Br	12	31	Si═X	19	8	P#Br	25	26	O%I
1	9	N−Si	8	25	C═Si	13	10	I═I	19	9	P#Si	25	31	O%X
1	10	N−I	8	26	C═I	13	15	I═X	19	10	P#I	26	4	S%S
1	15	N−X	8	31	C═X	15	31	X═X	19	15	P#X	26	5	S%Cl
1	19	O−O	9	2	N═N	16	17	C#C	19	23	F#F	26	6	S%P
1	20	O−S	9	3	N═O	16	18	C#N	19	24	F#Br	26	7	S%F
1	21	O−Cl	9	4	N═S	16	19	C#O	19	25	F#Si	26	8	S%Br
1	22	O−P	9	5	N═Cl	16	20	C#S	19	26	F#I	26	9	S%Si
1	23	O−F	9	6	N═P	16	21	C#Cl	19	31	F#X	26	10	S%I
1	24	O−Br	9	7	N═F	16	22	C#P	20	8	Br#Br	26	15	S%X
1	25	O−Si	9	8	N═Br	16	23	C#F	20	9	Br#Si	26	21	Cl%Cl
1	26	O−I	9	9	N═Si	16	24	C#Br	20	10	Br#I	26	22	Cl%P
1	31	O−X	9	10	N═I	16	25	C#Si	20	15	Br#X	26	23	Cl%F
2	4	S−S	9	15	N═X	16	26	C#I	20	25	Si#Si	26	24	Cl%Br
2	5	S−Cl	9	19	O═O	16	31	C#X	20	26	Si#I	26	25	Cl%Si
2	6	S−P	9	20	O═S	17	2	N#N	20	31	Si#X	26	26	Cl%I
2	7	S−F	9	21	O═Cl	17	3	N#O	21	10	I#I	26	31	Cl%X
2	8	S−Br	9	22	O═P	17	4	N#S	21	15	I#X	27	6	P%P
2	9	S−Si	9	23	O═F	17	5	N#Cl	23	31	X#X	27	7	P%F
2	10	S−I	9	24	O═Br	17	6	N#P	24	17	C%C	27	8	P%Br
2	15	S−X	9	25	O═Si	17	7	N#F	24	18	C%N	27	9	P%Si
2	21	Cl−Cl	9	26	O═I	17	8	N#Br	24	19	C%O	27	10	P%I
2	22	Cl−P	9	31	O═X	17	9	N#Si	24	20	C%S	27	15	P%X
2	23	Cl−F	10	4	S═S	17	10	N#I	24	21	C%Cl	27	23	F%F
2	24	Cl−Br	10	5	S═Cl	17	15	N#X	24	22	C%P	27	24	F%Br
2	25	Cl−Si	10	6	S═P	17	19	O#O	24	23	C%F	27	25	F%Si
2	26	Cl−I	10	7	S═F	17	20	O#S	24	24	C%Br	27	26	F%I
2	31	Cl-X	10	8	S═Br	17	21	O#Cl	24	25	C%Si	27	31	F%X
3	6	P−P	10	9	S═Si	17	22	O#P	24	26	C%I	28	8	Br%Br
3	7	P−F	10	10	S═I	17	23	O#F	24	31	C%X	28	9	Br%Si
3	8	P−Br	10	15	S═X	17	24	O#Br				28	10	Br%I
3	9	P−Si	10	21	Cl═Cl	17	25	O#Si				28	15	Br%X
3	10	P−I	10	22	Cl═P	17	26	O#I				28	25	Si%Si
3	15	P−X	10	23	Cl═F	17	31	O#X				28	26	Si%I
3	23	F−F	10	24	Cl═Br							28	31	Si%X
3	24	F−Br	10	25	Cl═Si							29	10	I%I
3	25	F−Si	10	26	Cl═I							29	15	I%X
3	26	F−I	10	31	Cl═X							31	31	X%X
3	31	F−X												

In carrying out genetic algorithm optimizations we have also made use of a "training set success measure", which differs from the "Briem and Lessel success measure" in that the molecular data set used is the training set, not the one defined by Briem and Lessel.

Basic Keysets Used. As a starting point we used both the 166 bit MDL keyset and the 960 bit MDL keyset. As noted earlier, the 960 bit keyset contains a number of keybits which can be set by more than one descriptor. If we remove all those multiply mapped keybits we are left with a 726 bit keyset. Alternatively, we can make each one of the descriptors encoded in the 960 bit keyset correspond to one unique keybit, yielding a 1387 bit keyset.

Finally, we can form a keyset by assigning a unique keybit to each of the descriptors encoded by the underlying key setting algorithm, assigning an occurrence count to each descriptor of "set 1 or more times".

Surprisal Calculations. The information theoretic concept of a surprisal is defined as[30]

$$I = -\ln(A/B)$$

where A and B are probabilities associated with observing the corresponding properties. Typically A is an experimentally observed probability and B is from a theoretically derived "prior" distribution. The surprisal thus provides a

Table 5. Custom and Sgroup Properties

n1	n2	n3	property
5	0	1	charge (in structure somewhere)
5	0	2	isotope
5	0	3	"other" atom type
5	0	4	CH_3
5	0	5	halogen
5	0	6	NH_2
5	0	7	five-membered ring
5	0	8	six-membered ring
5	0	9	Kekule-aromatic ring
5	0	10	seven-membered ring
5	0	11	eight-membered ring
5	0	12	103 < atom type < 256
5	0	13	more than one fragment
5	1	1−31	atom types 1−31
5	2	0−31	atom types 31−63
5	3	0−31	atom types 64−95
5	4	0−31	atom types 96−127
5	5	0−31	atom types 128−159
5	6	0−31	atom types 160−191
5	7	0−31	atom types 192−223
5	8	0−31	atom types 224−255
5	9	1	atom type 256
5	11	1	component Sgroup type
5	11	2	SRU Sgroup type
5	11	3	monomer Sgroup type
5	11	4	copolymer Sgroup type
5	11	5	alternating copolymer subtype
5	11	6	random copolymer subtype
5	11	7	block copolymer subtype
5	11	8	graft Sgroup type
5	11	9	formulation Sgroup type
5	11	10	mixture Sgroup type
5	11	11	cross-link Sgroup type
5	11	12	modification Sgroup type
5	11	13	any polymer Sgroup type
5	11	14	data Sgroup type
5 11 15 thru 5 13 5			data Sgroup field number
5	13	6	mer Sgroup type

symmetric measure of the degree to which the probabilities in the two distributions differ.

In this work we chose to associate A with the total probability for keybits to be set by molecules in the five active classes and similarly associate B with the probability for keybits to be set in the one inactive class, using the molecules in the Briem and Lessel data set.

Additionally, we noted that the frequency for a number of keybits was quite low, and so we also calculated the expected noise in the keybit frequency, assuming a Poisson distribution for the noise. This also allowed us to define a surprisal S/N ratio by dividing the calculated surprisal by the standard deviation of the noise

$$S/N = |I/(1/N_a + 1/N_b)^{1/2}|$$

where N_a and N_b are the keybit frequencies in the five active and one inactive classes, respectively.

Unlike the surprisal, this surprisal signal-to-noise ratio does depend on the size of the data sets; increasing the number of compounds increases the calculated S/N ratio. However, because of the relatively low frequency of some of the keybits, coupled with the small sample sizes, it was felt important to investigate the effects of explicitly accounting for statistical noise in these calculations.

Genetic Algorithm Optimization of Keysets. Use of genetic algorithms to optimize the choice of keybits was facilitated by use of the SUGAL Genetic Algorithm package.[31] SUGAL provided the underlying genetic algorithm methods; custom C code was written to handle the calculation of success measures for the proposed keys as well as some ancillary utility functions.

Table 6. Training Set Activity Classes

MDDR activity code	activity class
02452	TNF Inhibitor
09221	acetylcholinesterase inhibitor
12453	lipid peroxidation inhibitor
31430	angiotensin II blocker
31500	calcium channel blocker
54112	H+/K+-ATPase inhibitor
71522	reverse transcriptase inhibitor
71523	HIV-1 protease inhibitor
75721	aromatase inhibitor
78348	phospholipase A2 inhibitor
78351	lipoxygenase inhibitor
78362	thymidylate synthetase inhibitor
78371	collagenase inhibitor
78373	topoisomerase inhibitor
78374	protein kinase C inhibitor
78417	phosphodiesterase III inhibitor
78418	phosphodiesterase IV inhibitor

Training Set. Initial genetic algorithm optimization of keysets used the Briem and Lessel data set for both optimization and calculation of the final success measure. As should be expected, these GA optimizations, when run to completion, were found to be overtrained. We therefore constructed an independent data set to use in the GA optimizations.

This "training set" was culled from the MDDR database.[29] It consisted of 1700 compounds which belonged to one of 17 activity classes. Compounds which belonged to more than one activity class were removed from the data set. Each of the activity classes had more than 100 compounds, and so 100 were chosen at random from each activity class. Use of equal numbers of compounds in each activity class removed effects of differing class size from the "class-specific" success measure. Otherwise, classes with more members will tend to have higher success measures. Additionally, we did not use any of the activity classes used in the Briem and Lessel data set as a further step to orthogonalize the training and evaluation data sets. The activity classes used, and their MDDR activity codes, can be found in Table 6. A "training set success measure" could be calculated by substituting this training set for the Briem and Lessel data set in the success measure calculations outlined above.

RESULTS AND DISCUSSION

Random Pruning of the 3234 Bit Keyset. It is instructive to begin by considering the generation of keysets by random elimination of keybits from the 3234 bit keyset. Since each keybit in the 3234 bit keyset corresponds to a single descriptor, elimination of a keybit is equivalent to eliminating a descriptor from the keyset. Figure 1 shows the success measures for 96 such keysets, with sizes from 100 to 3200 keybits, which were generated from the 3234 bit keyset. There is noise in the resulting graph, but it is clear that the dependence of the observed success measure on the number of keybits is quite weak above about 1000 keybits in the keyset. Below that size there is a degradation in performance, which becomes noticeable by a keyset length of 500 keybits. Note, however, that even at lengths of 100 keybits the

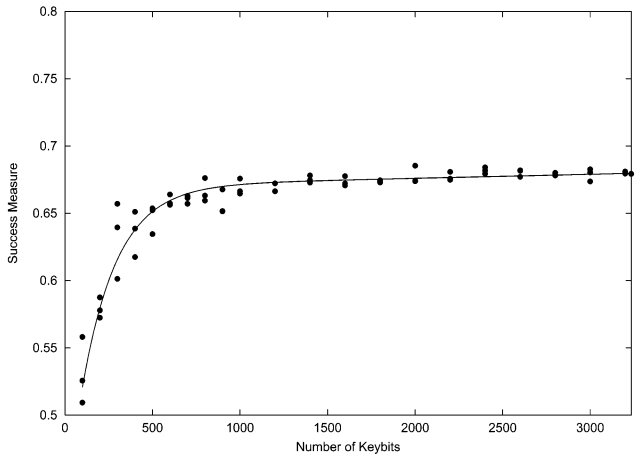

Figure 1. Plot of success measure as a function of keyset size for 96 randomly generated keysets derived from the 3234 bit keyset. Solid line is meant to guide the eye.

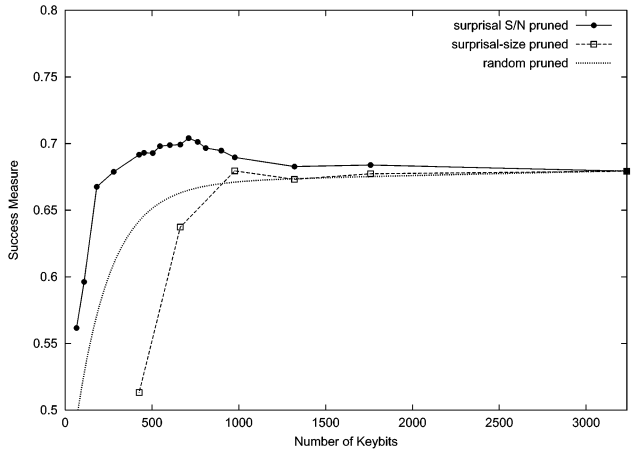

Figure 2. Plot of success measure as a function of keyset size for (a) randomly pruned 3234 bit keyset, (b) surprisal pruned 3234 bit keyset, and (c) surprisal S/N pruned 3234 bit keyset.

performance has only degraded by about 25%. Note also that the curve in Figure 1 is merely meant to guide the eye.

Surprisal Optimization of Basic Keysets. One might expect that a more effective pruning of keybits from a keyset could be obtained using the surprisal values or the surprisal S/N ratio for the various keybits as part of the selection criterion. Keybits with surprisals of zero are set at a statistically equal rate in both the composite of active classes and in the inactive class. This should make them less active/inactive differentiating than keybits which have a significantly different probability of being set in these two classes.

Figure 2 displays the results of pruning the 3234 bit keyset, using both surprisal and the surprisal S/N ratio as the selection criterion. In both cases keybits were eliminated with the lowest surprisal or S/N values, and the cutoff values were successively increased. For these calculations we used the Briem and Lessel data set to define the surprisals as well as in evaluating the success measure. Also plotted in Figure 2 are the results of the random pruning of the 3234 bit keyset.

It is evident that using the raw surprisal yields keysets which function significantly worse than those produced by random pruning. In contrast, the results of pruning based on the surprisal S/N ratio shows improvement over random pruning. There is even a broad maximum in the success measure at between 400 and 900 keybits. The maximum is

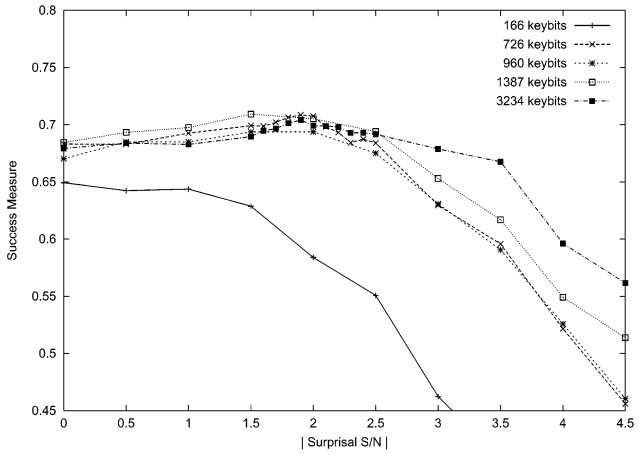

Figure 3. Plot of success measure as a function of the surprisal S/N threshold used in pruning the keysets.

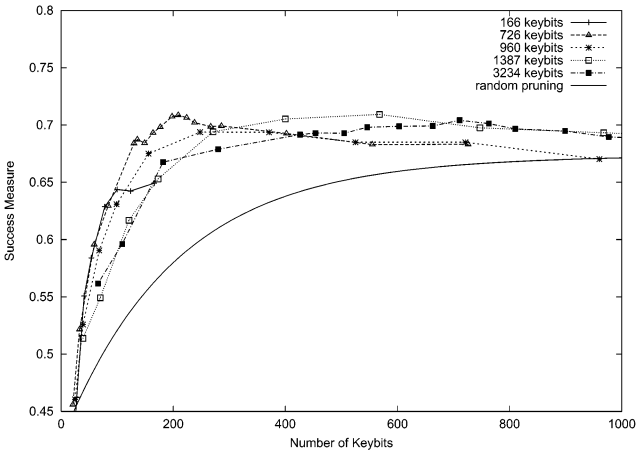

Figure 4. Plot of success measure as a function of the number of keybits in the pruned keyset.

located at a value of the surprisal S/N of 1.9, with 711 keybits used in the keyset, and a success measure of 0.704.

We next pruned the 166, 726, 960, and 1387 bit keysets described above. Figure 3 shows a plot of the success measures obtained as a function of the threshold surprisal S/N used in pruning the various keysets. Keysets were obtained by eliminating all keybits with a surprisal S/N less than the threshold. It is evident that the 166 bit keyset has inferior performance to the other keysets and that its performance is not improved by the surprisal S/N pruning. The other keysets show some improvement in performance as the pruning threshold is increased and appear to behave quite similarly as a function of the surprisal S/N. A broad maximum is found at a surprisal S/N of approximately 2, with degradation of performance as the threshold is raised further.

Figure 4 is an interesting recasting of this same data, this time as a function of keyset size. This plot shows the degree of commonality in performance between the keysets as a function of size. Most evident is the significant degradation in performance observed below 200 keybits. Interestingly, the performance of the 166 bit keyset lies on the curves for the 1387 and 3234 bit keysets.

The similar performance of the various pruned keysets seen in Figure 4 does not, however, imply commonality in the descriptors encoded in the keysets. Indeed, the overlap in common descriptors between the unpruned 166 bit keyset

Table 7. GA Results

initial				200 generations			400 generations		
av bits	mean success	success SD	max. success	bits	B.-L. success[a]	t. s. success[b]	bits	B.-L. success[a]	t. s. success[b]
194	0.499	0.016	0.540	454	0.698	0.676	538	0.698	0.687
258	0.522	0.013	0.555	475	0.710	0.676	548	0.711	0.687
323	0.536	0.011	0.573	505	0.702	0.675	580	0.703	0.686
485	0.555	0.008	0.582	636	0.708	0.678	648	0.705	0.687
646	0.565	0.007	0.588	708	0.701	0.677	742	0.701	0.685
970	0.576	0.005	0.586	949	0.707	0.674	938	0.705	0.683
1293	0.582	0.004	0.591	1147	0.705	0.671	1088	0.708	0.680
1617	0.585	0.003	0.593	1391	0.698	0.665	1342	0.706	0.676
av					0.704	0.674		0.705	0.684

[a] B.-L. success: Briem Lessel success measure. [b] t. s. success: training set success measure.

and either the pruned 1387 keyset (S/N threshold = 3.0, 173 keybits) or the 3234 keyset (S/N threshold = 3.5, 182 keybits) is ~16% and ~11%, respectively. However, they show very comparable success measures of 0.649, 0.652, and 0.668. The pruned 1387 and 3234 keysets have more overlap, but it is still only about 66%.

It is also interesting to note the behavior of the 726 bit keyset as it is pruned. The full keyset performs worse than the 1387 and 3234 bit keysets, but it exhibits a strong maximum in performance for small keyset size. At a S/N threshold of 1.9 it has a success measure of 0.708, and 208 keybits remaining in the keyset. This represents the best performance observed for surprisal S/N-based pruning of a keyset.

Genetic Algorithm Optimization of Keysets. We next turned to the use of Genetic Algorithm Optimization methods for construction of optimal keysets. Preliminary runs were carried out using the Briem and Lessel data set for both training and evaluation. Not surprisingly, the optimization produced highly overtrained keysets which had high success measures (up to 0.84) and abysmal performance on other problems. Further work was done using the alternate 1700 member training set described above in the optimization phase.

Table 7 presents results from a grid of calculations run starting with average fractions of 0.06, 0.08, 0.1, 015, 0.2, 0.3, 0.4, and 0.5 of the keybits in the 3234 bit keyset used. These runs used a population of 200 keysets. The initial success measures evaluated using the training set mirror the behavior seen in Figure 2, with success measures showing a very weak dependence on keyset size, with the dependence becoming stronger for very small keysets. Also evident is the fact that the distribution of performance across any of the populations is very narrow, with standard deviations of the distributions ranging from a high of 0.016 (or 3%) for the smallest keysets down to 0.003 (or 0.5%) for the largest. We used a scale normalization scheme with a bias value of 100 to try to mitigate the effects of these very narrow distributions.[31]

The Briem and Lessel and training set success measures for the optimized keysets show even less dependence on keyset size. We do note, however, that keysets which initially had fewer than an average of 700 keybits yielded optimized keysets with more keybits, and keysets with and initial average of more than 700 keybits yielded optimized keysets with fewer keybits. However, this trend was not strong enough in our GA calculations to point to an optimal keyset size.

Our initial calculations used a population of 200 keysets which was propagated for 200 generations. To test the quality of the convergence we extended these calculations for an additional 200 generations. As can be seen from the averages reported at the bottom of Table 7 the Briem and Lessel success measure changed by only 0.1%, suggesting that the calculations were well converged (at least locally) after 200 generations. The success measure evaluated using the training set did increase by 1.5%, presumably reflecting overtraining of the GA.

Overall, genetic algorithm optimization did produce keysets with better performance than our initial keysets. The best keyset had a success measure of 0.711, with a size of 548 bits. However, the average success measure of all the genetic algorithm optimized keysets was 0.704, slightly below the result for the best surprisal S/N pruned keyset.

CONCLUSIONS

We have demonstrated that it is possible to improve MDL keyset performance by reoptimizing keysets for use in molecular similarity. We have produced keysets with optimized success measures up to 0.711 versus the success measures of 0.649 and 0.670 for the standard MDL 166 and 960 bit keysets, which were constructed and optimized for substructure searching.

In performing these optimizations on MDL-based keysets we observed that increasing keyset size had little effect on overall performance for keysets larger than approximately 1000 bits. We also observed a striking similarity in performance between different keysets of similar size. This similarity in performance did not stem from the keysets being composed of the same descriptors, since descriptor overlap was generally low between the optimized keysets encountered here.

A number of candidate keysets were produced using random selection, surprisal-based selection, and surprisal S/N-based selection. Surprisal-based selection of keybits was found to underperform random selection, while surprisal S/N based selection was found to outperform random selection. The best performing keyset from this work contained 208 keybits and had a success measure of 0.708.

The great insensitivity of overall performance on both keyset size and identity of the descriptors encoded resulted in great difficulty in attempts to optimize the performance using standard mathematical techniques. Genetic Algorithm optimization is a particularly powerful tool, but while it succeeded in producing keysets with better performance, we

were unsuccessful in locating a single globally optimum keyset. Instead, we were able to produce families of well-performing keysets with defined starting keysets and defined, but variable, numbers of keybits in the initial, unoptimized key. Performance for the optimized keysets averaged 0.704, with the best keyset having a success measure of 0.711 and a length of 548 keybits.

This suggests that construction of optimized keysets needs to be driven by known constraints, such as our use of surprisal S/N, instead of autonomously, as is the case in our use of genetic algorithms. Future work is focusing on identification of data set-size independent constraints which can be used to direct the selection of descriptors used in keyset construction. Once such constraints can be identified we will also extend our work to include keysets constructed with occurrence counts different from "one or more".

ACKNOWLEDGMENT

We wish to acknowledge the help in this work afforded by the Cheshire development team, especially Richard Briggs. We also gratefully acknowledge conversations with Hans Briem and Uta Lessel and their help in obtaining the data sets used to calculate the success measure in this work.

REFERENCES AND NOTES

(1) Willett, P. In *Similarity and Clustering in Chemical Information Systems;* Wiley: New York, 1987; 266 pp.
(2) McGregor, M. J.; Pallai, P. V. Clustering of Large Databases of Compounds: Using the MDL "Keys" as Structural Descriptors. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 443−448.
(3) *Combinatorial Chemistry and Molecular Diversity in Drug Discovery;* Gordon, E. M., Kerwin, J. F., Ed.; Wiley: New York, 1998; 516 pp.
(4) Mason, J. S.; Hermsmeier, M. A. Diversity Assessment. *Curr. Opin. Chem. Biol.* **1999**, *3*, 342−349.
(5) Turner, D. B.; Tyrrell, S. M.; Willett, P. Rapid Quantification of Molecular Diversity for Selective Database Acquisition. *J. Chem. Inf. Comput. Sci.* **1997**, *37*, 18−22.
(6) Lipinski, C. A.; Lombardo, F.; Dominy, B. W.; Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug. Delivery Rev.* **1997**, *23*, 3−25.
(7) Teague, S. J.; Davis, A. M.; Leeson, P. D.; Oprea, T. The Design of Leadlike Combinatorial Libraries. *Angew. Chem., Int. Ed. Engl.* **1999**, *38*, 3743−3748.
(8) Oprea, T. I.; Davis, A. M.; Teague, S. J.; Leeson, P. D. Is There a Difference between Leads and Drugs? A Historical Perspective. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1308−1315.
(9) Henkel, T.; Brunne, R. M.; Müller, H.; Reichel, F. Statistical Investigation into the Structural Complementarity of Natural Products and Synthetic Compounds. *Angew. Chem., Int. Ed. Engl.* **1999**, *38*, 643−647.
(10) Shemetulskis, N. E.; Weininger, D.; Blankley, C. J.; Yang, J. J.; Humblet, C. Stigmata: An Algorithm To Determine Structural Commonalities in Diverse Datasets. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 862−871.
(11) Xue, L.; Stahura, F. L.; Godden, J. W.; Bajorath, J. Mini-fingerprints Detect Similar Activity of Receptor Ligands Previously Recognized Only by Three-Dimensional Pharmacophore-Based Methods. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 394−401.
(12) Xue, L.; Stahura, F. L.; Godden, J. W.; Bajorath, J. Fingerprint Scaling Increases the Probability of Identifying Molecules with Similar Activity in Virtual Screening Calculations. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 746−753.
(13) Brown, R. D.; Martin, Y. C. Designing Combinatorial Library Mixtures Using a Genetic Algorithm. *J. Med. Chem.* **1997**, *40*, 2304−2313.
(14) Koehler, R. T.; Villar, H. O. Design of Screening Libraries Biased for Pharmaceutical Discovery. *J. Comput. Chem.* **2000**, *21*, 1145−1152.
(15) Patterson, D. E.; Cramer, R. D.; Ferguson, A. M.; Clark, R. D.; Weinberger, L. E. Neighborhood Behavior: A Useful Concept for Validation of "Molecular Diversity" Descriptors. *J. Med. Chem.* **1996**, *39*, 3049−3059.
(16) Matter, H. Selecting Optimally Diverse Compounds from Structure Databases: A Validation Study of Two-Dimensional and Three-Dimensional Molecular Descriptors. *J. Med. Chem.* **1997**, *40*, 1219−1229.
(17) Rhodes, N.; Willet, P.; Dunbar, J. B., Jr.; Humblet, C. Bit-String Methods for Selective Compound Acquisition. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 210−214.
(18) Ahrens, E. K. F. Customization for Chemical Database Applications. In *Chemical Structures; The International Language of Chemistry;* Warr, W. A., Ed.; Springer-Verlag: Berlin, 1988; pp 97−111, and references therein.
(19) Christie, B. D.; Leland, B. A.; Nourse, J. G. Structure Searching in Chemical Databases by Direct Lookup Methods. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 545−547.
(20) MDL Information Systems, Inc., 14600 Catalina Street, San Leandro, CA 94577.
(21) Nagy, M. Z.; Kuzics, S.; Veszpremi, T.; Bruck, P. Substructure Search on Very Large Files using Tree-Structured Databases. In *Chemical Structures; The International Language of Chemistry;* Warr, W. E., Ed.; Springer-Verlag: Berlin, 1988; pp 127−130.
(22) Kauvar, L. M.; Higgins, D. L.; Villar, H. O.; Sportsman, J. R.; Engqvist-Goldstein, Å.; Bukar, R.; Bauer, K. E.; Dilley, H.; Rocke, D. M. Predicting ligand binding to proteins by affinity fingerprinting. *Chem. Biol.* **1995**, *2*, 107−118.
(23) Weinstein, J. N.; Myers, T. G.; O'Connor, P. M.; Friend, S. H.; Fornace, A. J., Jr.; Kohn, K. W.; Fojo, T.; Bates, S. E.; Rubinstein, L. V.; Anderson, N. L.; Buolamwini, J. K.; van Osdol, W. W.; Monks, A. P.; Scudiero, D. A.; Sausville, E. A.; Zaharevitz, D. W.; Bunow, B.; Viswanadhan, V. N.; Johnson, G. S.; Wittes, R. E.; Paull, K. D. An Information-Intensive Approach to the Molecular Pharmacology of Cancer. *Science* **1997,** *275*, 343−349.
(24) Briem, H.; Kuntz, I. D. Molecular Similarity Based on DOCK−Generated Fingerprints. *J. Med. Chem.* **1996**, *39*, 3401−3408.
(25) Lessel, U. F.; Briem, H. Flexsim-X: A Method for the Detection of Molecules with Similar Biological Activity. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 246−253.
(26) Briem, H.; Lessel, U. In vitro and in silico affinity fingerprints: Finding similarities beyond structural classes. *Perspect. Drug Discov. Design* **2000**, *20*, 231−244.
(27) Rarey, M.; Dixon, J. S. Feature trees: A new molecular similarity measure based on tree matching. *J. Comput.-Aided Mol. Design* **1998**, *12*, 471−490.
(28) Gushurst, A. J.; Nourse, J. G.; Hounshell, D.; Leland, B. A.; Raich, D. G.; The Substance Module: The Representation, Storage and Searching of Complex Structures. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 447−454.
(29) MACCS Drug Data Report, Release 2000.2, MDL Information Systems, Inc., 14600 Catalina Street, San Leandro, CA 94577, 2000.
(30) Levine, R.; Bernstein, R. B. In *Molecular Reaction Dynamics and Chemical Reactivity;* Oxford University Press: New York, 1987; p 260ff.
(31) Hunter, A., SUGAL Genetic Algorithm Package, http://osiris.sunderland.ac.uk/ahu/sugal/home.html, 1995.

CI010132R