A Heuristic Solution to the Functional Group Switching Problem in Organic Synthesis

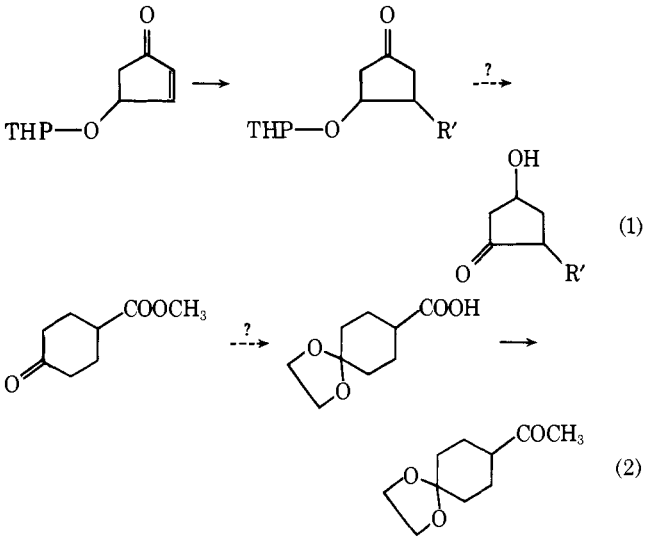
H. W. Whitlock

Contribution from the Department of Chemistry, University of Wisconsin, Madison, Wisconsin 53706. *Received April 10, 1975*

Abstract: A frequently encountered problem in organic syntheses is the functional group switching problem, e.g., how to reduce a methyl ester to a primary alcohol in the presence of a ketone elsewhere in the molecule. An efficient algorithm is presented for solution of this problem, the algorithm affording one or more routes that are among the set of shortest possible.

Introduction

This paper describes four programs written to explore heuristics for the functional group switching problem as defined below. These programs were written in Fortran IV and run on an IBM 7094 using the PUFFT[1] Fortran compiler. In devising a synthetic sequence to a desired compound, one frequently is confronted with the problem of manipulation of functional groups. This problem generally appears in one of two forms. Either one has effected a condensation reaction and must convert the functional groups that were necessary for the condensation into the desired functional groups, or one must modify the functional groups of a molecule in order to effect a desired condensation reaction. Examples are given in eq 1 and 2, respectively. We refer to this problem as the *functional group switching* (FGS) *problem*. It is a minor but recurring problem in organic synthesis. It recurs simply because one's primary interest is in constructing a carbon skeleton and one switches functional groups around as necessary to achieve this.



(1)

(2)

Any practically useful algorithm for the FGS problem is subject to several constraints. It must operate within reasonable time/space limitations for problems of reasonable complexity. Algorithm I discussed below is defective in this sense. If we think of the distance between two molecules as being related to the overall yield or number of synthetic steps linking them, we require that any FGS problem algorithm find a solution if one exists (we may place some upper limit on the acceptable distance) and that any solution found be of minimal distance. This is an important restriction since it immediately rules out approaches such as one having a predilection for explicitly defined blocking reactions.

The structural representation we have chosen for the FGS problem is as follows. If F1, F2, . . . , F*n* are functional groups and RGT is a reagent (see Tables I and II), then a reaction is an ordered triplet {RGT F1 F2}, meaning F1 —RGT→ F2. The set of all reaction triplets is the reaction dictionary. A "molecule" is an ordered set of functional groups [F1, F2, . . . , F*n*]. If a molecule [F1 F3] can be converted in one step into [F2 F4], and [F2 F4] can be converted in one step into [F5 F6], we say [F1 F3] ⇒ [F5 F6], "⇒" representing a possibly multistep synthetic sequence. In the above examples (eq 1 and 2) the FGS problems are

$$[\text{RCO RCHOTHP}] \overset{?}{\Rightarrow} [\text{RCHOH RCO}]$$

and

$$[\text{RCO COOMe}] \overset{?}{\Rightarrow} [\text{KETAL COOH}]$$

respectively (see Table II for functional group abbreviations).

Reactions with explicit qualifications such as "CH_2OH —TrCl→ CH_2OTR, CH_2OH is unhindered" are not present. Rather there is a functional group CH_2OH' that represents an unhindered primary alcohol. Structural qualifications are thus implicitly contained in the symbol of the functional group itself. Functional groups are isolated and do not interact. A C=C—COOMe part structure is either two isolated groups (C=C and CCOMe) or an entity in itself, but not both (see below). This has several important consequences. It greatly simplifies the FGS problem since the basic reaction dictionary triples {RGT F1 F2} are sufficient to define the reactions of all compounds. On the other hand, it is clearly a very abstract sort of molecular representation.

The reaction dictionary as defined above comprises a directed reaction graph wherein the nodes of the reaction graph are functional groups and there is an edge leading from node F1 to a node F2 if F1 —RGT→ F2 is in our reaction dictionary. The FGS problem for the case of a single functional group is then simply one of finding a path across the graph from the starting to target node. As we have discussed elsewhere,[2] the above-directed graph representation of a functional group reaction dictionary is in fact the transition graph of a finite automation. As such there are orderly procedures for constructing reaction graphs of compounds from them so that the FGS problem for a molecule is also one of finding a path across a directed graph from starting to target compound. The FGS problem is thus a decidable[3] one: the question is not "is a solution possible" but "is an efficient solution possible". That one must worry about the efficiency of solutions to the problem of finding a path across a reaction graph is clear when one considers that if there are *n* different functional groups and one is dealing with an *m*-ary FGS problem (*m* functional groups present), the compound reaction graph may have n^m nodes. Any solution that uses resources (run time or space) in a manner that depends linearly on the number of nodes in the graph being searched will quickly become unmanageable as *m* increases. If one thinks of an efficient solution as one wherein

Table I. Description of Reagents Used

Reagent name	No.	Description
OH	1	Sodium hydroxide
H_3O	2	Dilute aqueous acid
CH_2N_2	3	Diazomethane
LAH+	4	Lithium aluminum hydride Under vigorous conditions
$NaBH_4$	5	Sodium borohydride
LAH−	6	Lithium aluminum hydride at low temp
CrO_3Py	7	Basic Cr(VI) (Sarett's reagent)
CrO_3^+	8	Acidic Cr(VI) (Jones' reagent)
GLYCOL	9	Ethylene glycol under ketalization conditions
EVE	10	Ethyl vinyl ether, H^+
TsCl	11	Tosyl chloride/pyridine
NaBr	12	SN2 bromide ion
Ac_2O	13	Acetic anhydride/pyridine
RLi	14	Alkyllithium in excess
RMgX	15	Grignard reagent in excess

the amount of effort expended is linearly related to m, then it is almost certainly true[4] that no efficient solution to the FGS problem is possible. As we show below, however, one can devise heuristics that reduce the FGS problem to a manageable level of complexity for reasonably difficult problems (four to five functional groups, ten-step interconversions).

Rather than argue the merits of the above notation as a model of strucutre in general, we merely point out that it is a limited but useful model and must not be pushed too far.

Results

There are several common characteristics of the programs discussed below:

(1) The reaction dictionary is read as a set of triplets {RGT F*i* F*j*}. A reagent indexes a list of precursor-product functional group pairs and its print name. A functional group indexes only its print name. A compound indexes its structure [F1 F2 . . . F*n*] and various pieces of information such as synthetic precursors, heuristic distance to target, depth (distance) from starting material, and status (expanded or not expanded). There are initially only two (starting and target) compounds, new compounds being generated as the programs proceed.

(2) The value of the Fortran function FPROD (RGT,F1) is the functional group F2 such that F1 —RGT→ F2 (F1 = F2 if no reaction). The analogous function PROD(RGT CP1) for compounds returns as its value CP2 (CP1 —RGT→ CP2) and is based on FPROD in the following way:

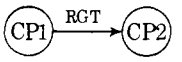
(a) Convert CP1 into an array C1(I),I = 1,N of the N functional groups.

(b) DO 1 I = 1, N

1 C2(I) = FPROD (RGT,C1(I))

(c) Turn the array C2(I), I = 1,N into the compound CP2.

In those algorithms that work backwards, there are the analogous functions FPRC (RGT,F1), the value of which is the functional group F2 such that F2 —RGT→ F1, and PRC (RGT,CP1). Since F2 is not normally unique when working backwards (COOMe —LAH→ CH_2OH, CHO —LAH→ CH_2OH), successive calls to PRC and FPRC are necessary for a given reagent. For a given FGS problem, the reaction graph is thus implicitly defined by the functions PROD and PRC. A function call CP2 = PROD (RGT,CP1) defines the part of the compound reaction graph:



while successive calls C*i* = PRC (RGT,CP1) for a given RGT produces

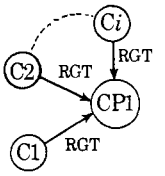


Table II. Description of Functional Group Symbols Used and Basic Reaction Dictionary for FGS Problems[a]

Functional group name	Group no.	Description	Reagent–product pairs[b]
CH_2OAc	1	Primary acetate	(OH CH_2OH) (LAH+ CH_2OH) (LAH− CH_2OH)
CH_2OH	2	Primary alcohol	(CrO_3Py) (CrO_3+ COOH) (EVE CH_2OEE) ($TsCl$ CH_2OTs) (Ac_2O CH_2OAc)
RCHOAc	3	Secondary acetate	(OH RCHOH) (LAH+ RCHOH) (LAH− RCHOH)
RCHOH	4	Secondary alcohol	(CrO_3PY RCO) (CrO_3+ RCO) (EVE RCHOEE) (TsCl RCHOTs) (Ac_2O RCHOAc)
COOMe	5	Methyl ester	(OH COOH) (LAH+ CH_2OH) (LAH− CH_2OH) (RLi R2COH) (RMgX R2COH)
COOH	6	Acid	(CH_2N_2 COOMe) (LAH+ CH_2OH) (RLi RCO)
ACETAL	7	Ethylene acetal	(H_3O CHO) (CrO_3+ COOH)
CHO	8	Aldehyde	(LAH+ CH_2OH) ($NaBH_4$ CH_2OH) (LAH+ CH_2OH) ($NaBH_4$ CH_2OH) (LAH− CH_2OH) (CrO_3+ COOH) (GLYCOL ACETAL) (RLi RCHOH) (RMgX RCHOH)
CH_2OEE	9	Primary ethoxyethyl ether	(H_3O CH_2OH) (CrO_3+ COOH) (GLYCOL CH_2OH)
RCHOEE	10	Secondary ethoxyethyl ether	(H_3O RCHOH) (CrO_3+ RCO) (GLYCOL RCHOH)
KETAL	11	Ethylene ketal	(H_3O RCO) (CrO_3+ RCO)
RCO	12	Ketone	(LAH+ RCHOH) ($NaBH_4$ RCHOH) (LAH− RCHOH) (GLYCOL KETAL) (RLi R2COH) (RMgX R2COH)
CH_2OTs	13	Primary tosylate	(LAH+ CH_3) (NaBr CH_2Br) (RLi CRUD) (RMgX CRUD)
CH_3	14	Methyl group	
CH_2Br	15	Primary bromide	(LAH CH_3) (RLi CRUD) (RMgX CRUD)
R2CHO	16	Tertiary alcohol	(CrO_3+ CRUD) (GLYCOL CRUD)
CRUD	17	Decomposition product	
RCHOTs	18	Secondary tosylate	(LAH+ CH_2) (RLi CRUD) (RMgX CRUD)
CH_2	19	Methylene group	

[a] The functional groups fall into eight equivalence classes under the relation of interconvertibility. They are [1,2,5,6,7,8,9], [3,4,10,11,12], [13], [14], [15], [16], [18], [19]. [b] For example the entry (OH CH_2OH) under functional group 1 says: CH_2OAc —OH→ CH_2OH.

Define a queue (initially empty) of compounds wherein a compound is added to the right end and removed from the left end of the queue.

STEP 1 Add startino compound to the queue. Go to STEP 2.

STEP 2 If product is on the queue then stop and trace the path to product from starting material. Otherwise go to step 3.

Step 3 Remove a compound (X) from the left end of the queue. Go to STEP 4.

STEP 4 Generate all products that can be formed in a single step from X. For each product as it is formed, see if it has been made so far. If so, do nothing and look at the next compound. If not, link it (with the reagent) to X and add it to the right end of the queue. Go to STEP 2.

Figure 1. Algorithm I: breadthfirst search algorithm for the functional group switching problem.

"Expansion" of a node, generation of all possible successors of it over all reagents, is achieved by:

$$\text{DO 1 } I = 1, \text{NRGTS}$$

$$\text{1 CP}(I) = \text{PROD}\,(I, \text{CP1})$$

(3) The problem of finding a synthetic path from starting to target compound is one of finding the shortest path from starting to target node on the implicitly defined reaction graph. In this sense the FGS problem is a very frequently recurring one in artificial intelligence.[5-8] Note that the reaction graph for an FGS problem can only be implicitly defined because of its very large size. For 20 different functional groups and a five functional group compound there may be $20^5 = 3\,200\,000$ nodes (compounds) on the reaction graph. A search tree (see Figure 5) is developed by nodewise expansion over all reagents starting with either the starting (when working forwards) or target compound (backwards). The search tree is some part of a spanning tree over the reaction graph and the synthetic route when found is a path through the search tree from starting to target compounds.

(4) As a measure of the efficiency of the search process we use the number of nodes expanded and generated during the search rather than run time. The idea of *depth* in a search tree is defined as the number of steps from the initial expanded node. Since, in working forwards, the starting node has a depth of zero, the depth of the target node is the length of the path found. The distance from starting to target compounds through a node, the through-distance of the node, is the depth of that node plus the node-target distance.

(5) The reaction dictionary used for the programs is in Table II. Reagents LAH+ and LAH− differ in that the former will reduce COOH, CH_2OTs, CH_2Br, and RCHOTs while the latter will not. CrO_3^+ will oxidize aldehydes and hydrolyze acetals and ketals while CrO_3Py will not. EVE is ethyl vinyl ether, while CH_2OEE is an ethoxyethyl ether and CRUD is a dead end functional group.

The simplest algorithm for finding a synthetic path from starting compound to target compound is one of breadth first search (Figure 1). This algorithm is guaranteed to produce the shortest route since the first in-first out nature of the queue forces this algorithm to expand nodes in the order in which they are generated. If one modifies the algorithm slightly so that all nodes of a given depth are expanded before formation of product is tested for, then *all* shortest routes will be produced. Algorithm I could clearly be run in either direction with no change in efficiency, but the forward search is simpler to program (function PROD vs. PRC). Since nodes are generated

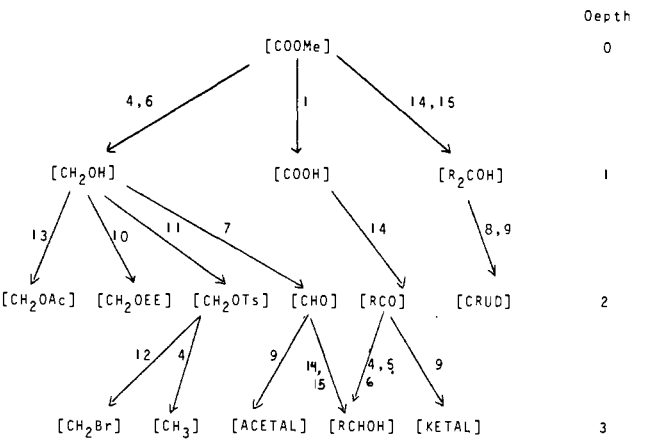
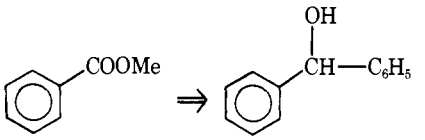


Figure 2. Search tree for [COOMe] ⇒ [RCHOH] employing Algorithm I. Reagents corresponding to the numbers are as in Table I. Ten nodes were expanded and 15 were generated out of the maximum of 19. Nodes [CH_2], [RCHOEE], [RCHOTs], and [RCHOAc] were not generated. The order of expansion of the nodes was: [COOMe], all nodes at depth 1 from left to right, all nodes at depth 2 from left to right.

as the program proceeds, only some fraction of the reaction graph's spanning tree is generated although, in the case wherein the synthesis cannot be achieved, the entire spanning tree will ultimately be generated. Figure 2 shows the search tree for the FGS problem

$$\text{[COOMe]} \stackrel{?}{\Rightarrow} \text{[RCHOH]}$$

as in



The reagent sequence for this FGS problem is (1) LAH+ or LAH−, (2) CrO_3Py, (3) RMgX or RLi; or (1) OH, (2) RLi, (3) $NaBH_4$ or LAH+ or LAH−.

The term breadth first alludes to the fact that nodes are expanded on a first produced-first expanded basis. As a result all nodes at depth 0 are expanded, then all nodes at depth 1, etc. The search tree thus grows evenly out from the starting node.

For the FGS problem [CHO COOMe] ⇒ [CHO CH_2Br] 46 nodes were expanded with generation of 83 compounds with the following six paths produced.

GLYCOL LAH− TsCl NaBr H_3O

GLYCOL LAH+ TsCl NaBr H_3O

GLYCOL LAH− TsCl H_3O NaBr

GLYCOL LAH+ TsCl H_3O NaBr

GLYCOL LAH− H_3O TsCl NaBr

GLYCOL LAH+ H_3O TsCl NaBr

Longer paths such as

$NaBH_4$ EVE LAH+ TsCl NaBr H_3O CrO_3Py

that also do the trick were not produced. There are incidentally an infinite number of (uninteresting) solutions to this FGS problem.

Algorithm I is unacceptable. It is so for the reason that *any* exhaustive search procedure is unacceptable. The search space is too large, and for any but the shortest solutions or simplest problems the available computational resources will be exhausted. For example, in trying to solve the FGS problem

Define a list of compounds to be expanded (initially empty) and starting (STRT) and target (TRGT) compounds. Each compound has associated with it two values: DEPTH, the distance (number of synthetic steps) to it from STRT; and DISTANCE, the estimated distance from it to TRGT. The sum of depth and distance for a compound is thus the estimated distance from STRT to TRGT through it as an intermediate. Although the distance of a compound may be computed for any compound (by IDIST (CPD, TRGT)), its depth is defined only when it is generated as a product derived from STRT.

STEP 1 Set the depth of STRT to 0 and add it to the list of compounds to be expanded. Go to STEP 2.

STEP 2 If TRGT is on the list then stop and trace the path to TRGT from STRT. Otherwise go to STEP 3.

STEP 3 a) Find that set of compounds on the list that are on the shortest estimated path to TRGT, i.e. they have the shortest sum depth plus distance. If this list is empty or the estimated through-distance of this "best" set is too big, quit with failure, otherwise

b) from this set pick one compound (X) that has the maximum depth. Although the estimated distance from STRT to TRGT through X is the same as other members of this best set X is presumably closest to TRGT. Go to STEP 4.

STEP 4 Generate all products that can be formed from X in a single step. For each product as it is generated see if it has been made so far If so do nothing and look at the next product. If not link it (with the reagent) to X, set its depth to DEPTH(X)+1 and add it to the list. When finished go to STEP 2.

Figure 3. Algorithm II: depthfirst search with a distance estimator.

$$[CH_2OAc\ COOMe\ CH_2OH] \Rightarrow [CH_2OH\ COOMe\ CH_2OAc]$$

the program bombed out by consumption of available space after approximately 300 compounds had been generated. A solution such as

$$\{CrO_3Py\ OH\ CH_2N_2\ EVE\ NaBH4\ AC20\ H30\}$$

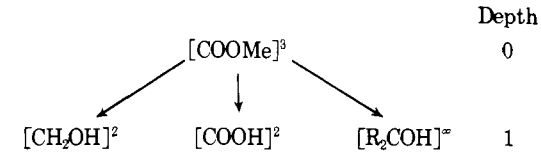
which is easily arrived at by use of the heuristics discussed below was not found. In fact the only thing this algorithm is really good for is to illustrate the general approach and to point up the necessity of heuristics for solution of the FGS problem.

Now note that algorithm I expands nodes on a first generated-first expanded basis. Suppose that we have a function IDIST (CPD, TRGT) that returns as its value an estimate of the distance (number of steps) from compound CPD to the target compound TRGT, and that we modify algorithm I to algorithm II (Figure 3).

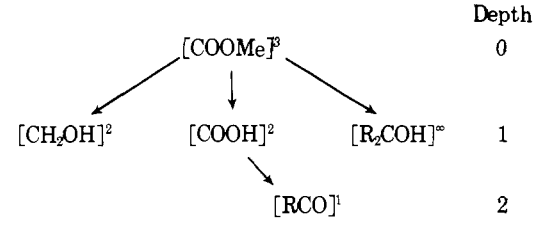
Algorithm II is essentially the famous Dijsktra algorithm for graph search[9] employing depthfirst search[10] and works in the following way. Step 1 sets the list of compounds to be expanded to contain only one member, the starting compound STRT. One then cycles through steps 2, 3, and 4 in that order until either the target compound is found (successful termination, step 2) or failure occurs (step 3a). Each cycle of steps 2, 3, and 4 results in the generation of all successors of some compound.

We may illustrate the operation of algorithm II by considering the above FGS problem [COOMe] ⇒ [RCHOH]. For this problem STRT is [COOMe] and TRGT is [RCHOH]. Let us assume that our distance-calculating function is perfect; i.e., the distance of [COOMe] is 3 ([COOMe] → [CH_2OH] → [CHO] → [RCHOH]), the distance of [R_2COH] is ∞, etc. Application of steps 1, 2, 3, and 4 results in production of three successors of [COOMe] ([CH_2OH], [COOH], and [R_2COH]) with the search tree looking like that in Scheme I, wherein the superscripts are the distances to product. Note that the list of compounds to be expanded is just a list of the "leaves" of the search tree.

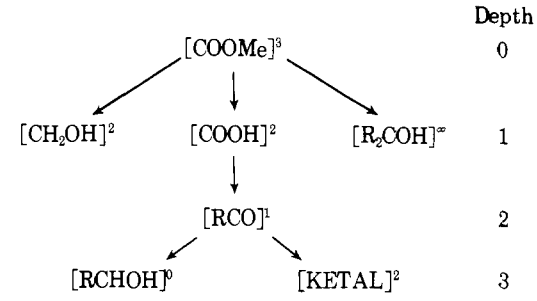
Scheme I



Scheme II



Scheme III



We now apply step 2, but since TRGT is not on the list we go to step 3. In step 3a we choose the best set {[CH_2OH], [COOH]} and in step 3b we chose (arbitrarily, since the depth of [CH_2OH] and [COOH] is the same) [COOH]. In step 4 we expand [COOH], generating all of its synthetic successors. The search tree now looks like that in Scheme II.

We now loop to step 2. TRGT is not on the list, so we go to step 3. In step 3a we choose the best set {[CH_2OH] [RCO]} whose sums distance and depth are a minimum (3); in step 3b we choose that member [RCO] of the best set whose depth is greatest (2 vs. 1), and in step 4 we expand [RCO] to produce Scheme III.

Looping to step 2 we find TRGT and quit with a synthetic route [COOMe] → [COOH] → [RCO] → [RCHOH]. Comparison of the final search tree with that in Figure 2 shows the simplification that results from using a distance function. Since one branch of the search tree may grow much faster than the others we refer to this as a depth first search process, in contrast to the breadth first approach involving even growth of all branches. Clearly, if the rapidly growing branch is in fact growing toward TRGT this is a more efficient way to proceed.

Now the central question surrounding algorithm II deals with the nature and accuracy of our distance function IDIST since in the absence of this function algorithm II is essentially the same as algorithm I. The function IDIST can be of one of several forms:

(1) It may be perfect in that its distance estimate is exactly the distance of the shortest path from CPD to TRGT. In this case, as above, we always pick the right compound in step 3 and go directly to TRGT, expanding n nodes for an n-step shortest path. We can think of no perfect function for IDIST. Application of algorithm I would of course give one a perfect distance function, but the cure and the disease would be indistinguishable in this case.

(2) IDIST might be erratic and occasionally overoptimistic (IDIST (CPD,TARGET) greater than the actual CPD-TARGET distance). In this case, there is no guarantee of

finding the *shortest* route from starting to target compound, as with an overly optimistic function one could easily go chasing down a depth first search and end up with a too long solution. Although this type of function could substantially narrow the search space, we reject it on the grounds that although any one solution is acceptable, it *must* be a shortest path.

(3) IDIST might be erratic but always conservative (IDIST (CPD, TARGET) less than or equal to the actual CPD-TARGET distance). In this case any path found is guaranteed[6,9] to be a shortest path although, depending on the accuracy of IDIST, one may go up a number of blind alleys before the shortest path is found.

The problem thus boils down to finding the most accurate conservative heuristic distance function. Algorithm II must also be modified (in step 4) when an imperfect distance estimator is employed since one may find a shorter route to a compound previously generated.[9,11] This is just a matter of bookkeeping (if it was previously generated, see if the new route to it is shorter and if so link it to X rather than its old predecessor and reset its depth) and does not represent a major change in the algorithm.

The question of a heuristic distance function is clearly equally applicable to working both forwards and backwards. There appears to be no reason to favor either direction, so for ease of programming the forward expansion is probably preferable.

Heuristics

(1) The simplest and easiest to calculate (and least useful) heuristic is: if any two functional groups in a compound are the same and the corresponding pair in the target compound are different, the conversion cannot be achieved. For example,

$$[\text{COOMe COOMe}] \not\Rightarrow [\text{CH}_2\text{OH COOMe}]$$

This follows from our concept of a functional group. A logical function IMPSBL (CPD,TRGT) (or IMPSBL (STRT,CPD) when working backwards) achieves this. This is an important heuristic if for no other reason than that it keeps one from expanding nodes that cannot give the target compound. Except for this though it does not narrow the search space at all since if one thinks of it as returning two distance values (∞ or a constant) algorithm II will be reduced to a breadthfirst search over the not impossible nodes, essentially as in algorithm I.

(2) For an FGS problem

$$[\text{FG1 FG2} \ldots \text{FG}N] \overset{?}{\Rightarrow} [\text{FP1 FP2} \ldots \text{FP}N],$$

break it down into N unary subproblems

$$[\text{FG1}] \overset{?}{\Rightarrow} [\text{FP1}]$$
$$\vdots$$
$$[\text{FG}N] \overset{?}{\Rightarrow} [\text{FP}N]$$

and apply a perfect distance estimator based on algorithm I to each subproblem. This will produce a set of N distances, the *maximum* of which is the *minimum* distance from starting to target compound; i.e., the difficulty of an FGS problem is at least that of its most difficult subproblem. This is clearly practical since for 20 functional groups and $N = 5$, one goes from a 20^5 (3 200 000) node reaction graph to five 20-node graphs for the five subproblems. For example, for the FGS problem

$$[\text{CHO COOMe}] \overset{?}{\Rightarrow} [\text{CHO CH}_2\text{Br}]$$

the subproblem [CHO] ⇒ [CHO] has a distance of 0 and the subproblem [COOMe] ⇒ [CH_2Br] has a distance of 3. There are *at least* three steps necessary for this FGS problem, but there may be more. This is clearly a conservative function since it will never underestimate the distance between two nodes and,

1. Set CTR to 0. Define a list L having one member, the pair: [N-th functional group of target molecule; CTR]. Go to 2.
2. Set pointers P and Q to point to L (P points to the functional group being expanded and Q points to the last member of L). Go to 3.
3. Set X to the functional group at P. Go to 4.
4. For every reagent RGT do the following:
   Generate all precursor functional groups Y by the function PRC (RTG, X) (Y $\xrightarrow{\text{RGT}}$ X). For each Y as it is produced see if it is contained in L. If so do nothing, otherwise add the pair (Y CTR+1) to the right end of the list (at Q) and move Q to the right. When done for all reagents, go to 5.
5. If P=Q then quit. L will point to a list ((Y1 1)----[Yi, M)) where M is the distance of the ith functional group to the target. Otherwise go to 6.
6. Move P one member to the right. Set CTR to the value contained at P. Go to 3.

Figure 4. Algorithm III: heuristic distance generator.

although it is perfectly accurate for the unary case, its accuracy will drop off as the number of functional groups increase. The distance predictor is precomputed as in algorithm III (Figure 4) and is used in a table lookup manner in the following way. For each distinct functional group in the target (assuming we are working forwards) algorithm III is applied to produce a list of functional group-distance pairs that can be its precursors. For the above FGS problem [CHO COOMe] ⇒ [CHO CH_2Br] the two lists:

((CHO 0)(ACETAL 1)(CH_2OH 1)(CH_2OAc 2)(COOMe 2)(COOH 2)(CH_2OEE 2))

and

((CH_2Br 0)(CH_2OTs 1)(CH_2OH 2)(CH_2OAc 3)(COOMe 3)(COOH 3)(CHO 3)(CH_2OEE 3))

are produced. The function IDIST (CPD TRGT) merely looks up each functional group of CPD on the appropriate precursor list and returns the maximum distance found. If a functional group is not found on the list the conversion is impossible (the function IMPSBL is accordingly dispensed with). Note that these precursor lists are independent of the FGS problem under consideration. If the reaction dictionary is static, one merely needs to load the appropriate previously calculated precursor lists as data. Moreover one can apply this heuristic in either direction, using product lists (compounds made from starting material) when working backwards.

The use of this heuristic substantially narrows the search space. For the FGS problem [CHO COOMe] ⇒ [CHO CH_2Br], one goes from having to expand 46 nodes (algorithm I with IMPSBL) to expansion of only 12 nodes with this heuristic. The answer is of course the same, namely (GLYCOL, LAH^+, TsCl, H_3O, NaBr).

(3) Since heuristic 2 is a perfect distance estimator for a unary FGS problem, occupies little space, and considerably narrows the search space for larger FGS problems, there is obviously much to be gained by expanding this heuristic to cover all *pairs* of functional groups in a compound. We do the same as above but define all functional group pairs that are precursors of the corresponding pairs in the target molecule. For the FGS problem [CHO COOMe CH_2OH] ⇒ [CH_2OAc COOMe CH_2OH] there are constructed precursor lists of the pairs [CH_2OAc COOMe], [CH_2OAc CH_2OH], and [COOMe CH_2OH]. Now this leads to a considerable expansion of the amount of storage space necessary. For 20 different

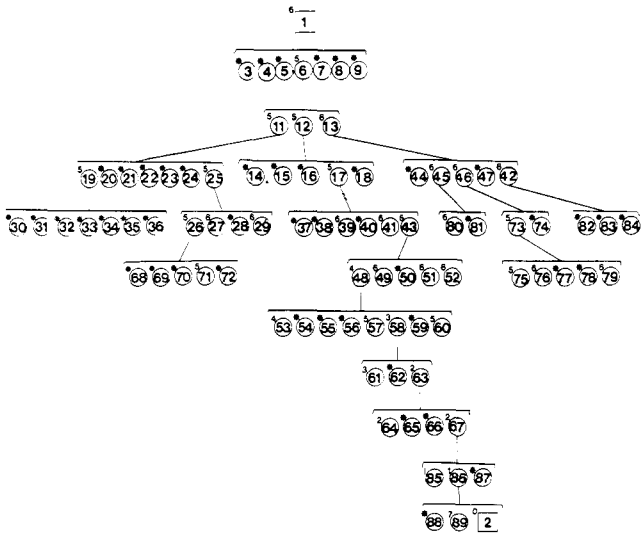


Figure 5. Search tree for the FGS problem [CH_2OH CH_2OAc COOMe CH_2OEE] ⇒ [CH_2OAc CH_2OH COOMe CH_2Br]. The starting node is **1**, the target node is **2.** The superscript (integer or asterisk) of each node is its heuristic distance to target, an asterisk signifying that the node cannot be a precursor of the target (distance = ∞). The nodes are numbered in order of their generation. The order of expansion of the nodes was: **1, 6, 12, 11, 25, 19, 17, 13, 43, 48, 58, 63, 26, 46, 73, 45, 42, 67, 86.**

functional groups and a five-functional-group compound, there will be up to ten lists, each with a maximum of 400 (20^2) precursor pairs. This is still manageable although expanison to all triplets would not be (ten lists, each having a maximum of 8000 members (20^3) for the above example).

Calculation of these precursor pairs is done exactly as in algorithm III. The resulting distance estimator is, of course, perfect for a 2-ary FGS problem, [CHO COOMe] ⇒ [CHO CH_2Br] being solved with the expansion of the minimal number of nodes (4, as compared to 12 and 46 for the two weaker heuristics).

This heuristic is now sufficiently powerful that one can solve problems of substantial complexity. A number of purposefully complicated examples are contained in Table III. It is interesting that it is rather difficult to devise solvable FGS problems that are sufficiently difficult to stretch the program embodying this heuristic. It is fairly clear why heuristic 3 is much more powerful than heuristic 2. Heuristic 2 merely asks itself "how can I convert F1 into F2" while heuristic 3 asks "how can I turn F1 into F2 and at the *same time* turn F3 into F4". The latter is much better at dealing with FGS subproblems wherein one has interfering functional groups.

The search tree of a relatively complicated FGS problem solved by this program is in Figure 5 and illustrates several features of the search. Node **1** is starting material [CH_2OH CH_2OAc COOMe CH_2OEE], and node **2** is the target, [CH_2OAc CH_2OH COOMe CH_2Br]. The problem is to convert an ethoxyethyl ether into a bromide and at the same time switch an alcohol and acetate, leaving a carbomethoxy group uneffected. Compounds are numbered sequentially as they are first generated. Compound **42** appears out of order. It was first generated as a successor of compound **17** via the route **1** —CrO_3Py→ [CHO CH_2OAc COOMe CH_2OEE] (**6**) —H_3O→ [CHO CH_2OAc COOMe CH_2OH] (**12**) —TsCl→ [CHO CH_2OAc COOMe CH_2OTs] (**17**) —Glycol→ [ACETAL CH_2OAc COOMe CH_2OTs] (**42**). Later in expansion of **13,** a shorter route was found: **1** —CrO_3Py→ **6** —Glycol→ [ACETAL CH_2OAc COOMe CH_2OH] (**13**) —TsCl→ **42.** Inspection of Figure 5 shows the consequences of the node-to-expand choice rules (Figure 3). The algorithm looks at all nodes of a given through-distance (depth plus heuristic distance to target) before it will try nodes of a larger through-distance.

Table III. Comparison of FGS Problems Using Unidirectional and Bidirectional Search Methods[a]

Case	Starting compd	Targent compd	Unidirectional		Bidirectional		Steps	Representative solution sequences
			Generated	Expanded	Generated	Expanded		
1	[RCHOEE RCO COOH]	[RCHOEE RCO RCHOH]	103	29	82	22	8	(GLYCOL EVE RLi $NaBH_4$ Ac_2O H_3O EVE OH)
2	[CH_2OH RCO COOH]	[CH_2OH RCO R_2COH]	35	4	36	4	4	(GLYCOL RLi RMgX H_3O)
3	[CH_2OH COOH]	[CH_2OH R_2COH]	17	2	15	2	2	(RLi RLi) or (RLi RMgX)
4	[CH_2OH COOMe]	[COOMe CH_2OH]	20	3	21	3	3	(CrO_3+ LAH – CH_2N_2)
5	[CH_2OH COOH]	[COOH CH_2OH]	19	3	19	3	3	(CH_2N_2 CrO_3 + LAH –)
6	[CH_2OH CH_2OAc COOMe CH_2OEE]	[CH_2OAc CH_2OH COOMe CH_2Br]	89	18	99	23	10	(CrO_3Py H_3O TsCl NaBr OH EVE $NaBH_4$ Ac_2O CH_2N_2 H_3O)
7	[CH_2OAc CH_2OH COOMe]	[CH_2OH CH_2OH COOH]	9	1	28	2	1	(OH)
8	[CH_2OH CH_2OAc CH_2OEE]	[CH_2OAc CH_2OH CH_2Br]	86	19	62	17	9	(CrO_3Py H_3O TsCl OH EVE $NaBH_4$ Ac_2O H_3O NaBr)
9	[CH_2OH CH_2OH COOH]	[CH_2OAc CH_2OH COOMe]	2	0	12	2		
10	[CH_2OH CH_2OAc COOMe CH_2OEE]	[CH_2OH CH_2OAc COOME CH_2Br]	28	5	27	5	5	(CrO_3Py H_3O TsCl NaBr $NaBH_4$)
11	[CHO COOMe CH_2OAc]	[CH_2OAc COOMe CHO]	37	10	48	10	7	($NaBH_4$ EVE OH CrO_3Py CH_2N_2 H_3O Ac_2O)
12	[CHO COOMe]	[CH_2OAc CH_2Br]	28	7	28	7	7	(GLYCOL LAH+ TsCl NaBr H_3O $NaBH_4$ Ac_2O)
13	[RCO COOH]	[RCO RCHOH]	18	4	20	4	4	(GLYCOL RLi $NaBH_4$ H_3O)
14	[CH_2OH CH_2OAc CH_2OEE]	[CH_2OAc CH_2OEE CH_2OH]	66	17	189	135		

[a] Columns 2 and 3 are starting and target compounds, respectively. Columns 4 and 5 are the number of compounds generated and expanded in the unidirectional heuristic 3 program. Columns 6 and 7 are likewise for the bidirectional search. Column 8 is the minimum number of steps necessary for the conversion.

Table IV.

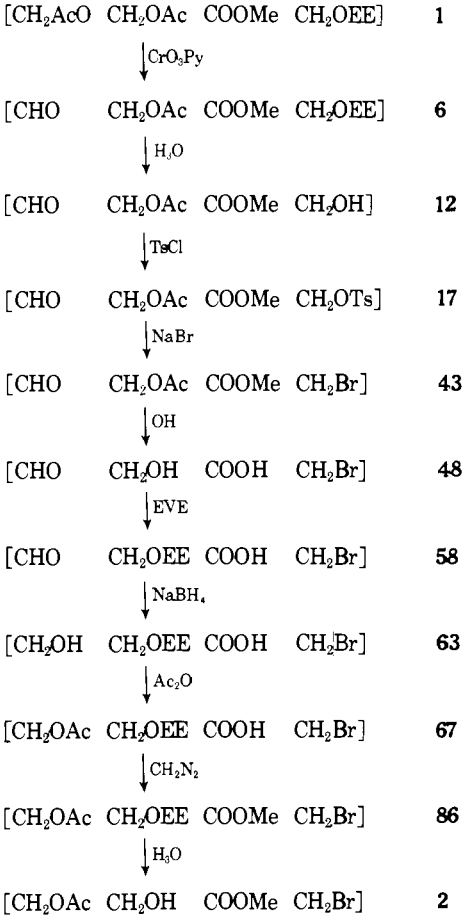
Node	Structure	Depth	Heuristic dist to 2
2	[CH_2OAc CH_2OH COOMe CH_2Br]	10	0
26	[CHO CH_2OAc COOH CH_2OH]	4	5
63	[CH_2OH CH_2OEE COOH CH_2Br]	7	2
67	[CH_2OAc CH_2OEE COOH CH_2Br]	8	2

It is instructive to consider why it is that the program (correctly) decided to expand node **63** rather than **26** but then, rather than expand the successor node **67** (see Figure 5), went back and expanded node **26,** thereby missing a quick "win". See Table IV. Node **63** is chosen for expansion rather than node **26** for the following reasons. The heuristic distance from **26** to **2** is five, this being the number of steps necessary to solve the most difficult **26** ⇒ **2** subproblem: [CHO CH_2OAc] ⇒ [CH_2OAc CH_2OH] via (OH EVE $NaBH_4$ Ac_2O H_3O). Similarly, the heuristic distance from **63** to **2** is two, this being determined by the three equally difficult subproblems: [CH_2OH CH_2OEE] ⇒ [CH_2OAc CH_2OH]; [CH_2OH COOH] ⇒ [CH_2OAc COOMe]; and [CH_2OEE COOH] ⇒ [CH_2OH COOMe], each of which requires two steps. Although the through-distances (depth + heuristic distance) of **26** and **63** are the same (nine), node **63** is chosen for expansion because it is at the greater depth. Note that the heuristic distance estimator is inaccurate: the *actual* distance from **63** to **2** is three steps rather than two, as is apparent on combining the three two-step binary subproblems into the ternary **63** ⇒ **2** subproblem [CH_2OH CH_2OEE COOH] ⇒ [CH_2OAc CH_2OH COOMe] that requires three steps (Ac_2O H_3O CH_2N_2).

Now node **67** is produced as a successor of **63** and what one would like to have happen is expansion of **67** to produce **86** followed by expansion of **86** to produce **2**. Instead node **26** was chosen for expansion rather than node **67** and the program wasted time and space in expanding nodes (**26, 46, 73, 45,** and **42**) in that order before it came back to **67.** Why? The heuristic distance from **67** to **2** (two) is accounted for by the subproblem [CH_2OEE COOH] ⇒ [CH_2OH COOMe]. Since **67** is a successor of **63** its depth is one greater and its calculated through-distance is ten steps. But node **26** has a through-distance of nine, is accordingly a better bet for expansion than **67,** and is so chosen. This occurs because in this case the heuristic distance of **67** equals the actual distance to **2** while the distance of **26** (by definition a minimal estimate) is in fact less than the actual distance. Thus we see that this type of behavior is a charactersistic of the heuristic approach used.

It is apparent from Figure 5 that the more inaccurate the heuristic distance function, the deeper the search tree will be developed in a breadthfirst manner before the advantages of the depthfirst search are realized. The final sequence from this search tree is shown in Scheme IV. Inspection of this sequence reveals a rather subtle use of "blocking" reactions. The program first solves the subproblem [CH_2OH CH_2OEE] ⇒ [CHO CH_2Br] (**1** ⇒ **43**) wherein the primary alcohol is blocked by conversion to an aldehyde and the bromide is then formed. The subproblem [CHO CH_2OAc] ⇒ [CH_2OAc CH_2OH] (**43** ⇒ **2**) is then solved, during the course of which a primary alcohol is reversibly blocked as its ethoxyethyl ether. It is important to realize that these blocking reactions are a *result* of the algorithm rather than an aspect of its operation. Algorithm II relies exclusively on a heuristic distance function rather than blocking reactions. That the result contains various blocking-deblocking sequences is merely a reflection of their being necessary for the shortest solution to the problem. It seems unlikely, in fact, that performance at this level could be achieved with an algorithm that operated on a blocking-deblocking basis.

Scheme IV



(4) Pohl[11] has advocated the strategy of bidirectional search as the most efficient way of finding the shortest path across a graph. As the name suggests one works forwards *and* backwards at the same time, the two search trees meeting somewhere in the middle of the search space. The argument is a little vague but the decrease in search space is expected to be roughly a factor of 2 ($2\pi(r/2)^2$ vs. πr^2 for two-space, where r is the node-node distance of interest). To see if the above programs could be improved further in this manner, we rewrote that one embodying heuristic 3 to work both forwards and backwards. The same heuristic based on functional group pairs was used although now one has precursor pair *and* product pair lists. As anticipated, there was no gain in efficiency. If anything, unidirectional heuristic 3 has a slight edge on bidirectional heuristic 3, although the difference is small (see Table III). This may be traced to the depthfirst search procedure used. Once a node is chosen whose heuristic through-distance (depth plus heuristic distance to target or starting material) is equal to the actual through-distance, the depthfirst procedure will go straight to target. Only in the case of a very inaccurate conservative heuristic should the bidirectional search prove the more effective.

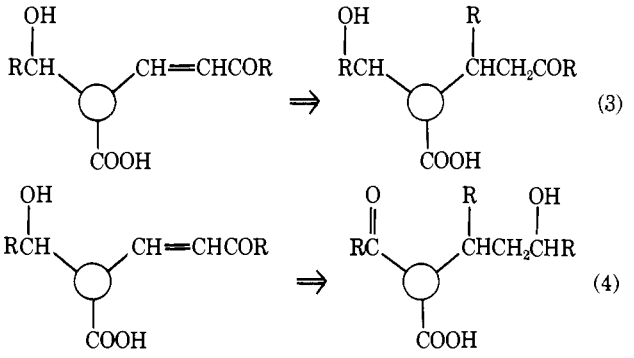
Extension to Larger Functional Arrays

We do not view the basic functional group definition as too limiting to be useful. Consider the comment above that an α,β-unsaturated ketone must be treated as an entity. Table V contains a substantial subset of the reactins of an α,β-unsaturated ketone (generated manually). Each part structure therein may be viewed exactly as a functional group. With addition of the reagents NaOOH (alkaline hydrogen peroxide) and RCu to the reaction dictionary in Table II and combina-

Table V. Symbols, Descriptions and Reagent–Product Pairs of Enone-Derived Functionalities (e.g., 2-Cyclohexenone)

Enone derivatives	No.	Description	Reagent–product pairs[a,b]
C=C–CO	F1	α,β-Unsaturated ketone (basic structure)	(4 F2) (5 F2) (6 F2) (9 F3) (14 F4) (15 F4) (16 F5) (17 F6)
C=C–CHOH	F2	Allylic secondary alcohol	(7 F1) (8 F1) (10 F7) (13 F8) (9 CRUD) (11 CRUD)
C=C–KETAL	F3	Ethylene ketal of F1	(2 F1) (8 F1)
C=C–C(R)OH	F4	Tertiary allylic alcohol	(8 CRUD) (9 CRUD) (11 CRUD)
O–C–C–CO	F5	Epoxide of F1	(6 F15) (14 F11) (15 F11) (4 F9) (5 F9) (9 F10)
RC–CHCO	F6	Dialkyl cuprate product from F1	(9 F12) (4 F13) (5 F13) (6 F13) (14 F14) (15 F14)
C=C–CHOEE	F7	Ethoxyethyl ether of F2	(2 F2) (9 F2) (8 F1)
C=C–CHOAc	F8	Acetate of F2	(1 F2) (4 F2) (5 F2) (14 F2) (15 F2)
HOC–CH–CHOH	F9	1,3-Diol (from F5 and LAH+)	(7 CRUD) (8 CRUD) (11 CRUD) (13 F16) (10 F17)
O–C–C–KETAL	F10	Ethylene ketal of F5	(2 F5) (8 F5)
O–C–C–C(R)OH	F11	Tertiary alcohol from F5 and RMgX	(8 CRUD) (9 CRUD) (11 CRUD)
RC–CH–KETAL	F12	Ethylene ketal of F6	(2 F6) (8 F6)
RC–CH–CHOH	F13	Secondary alcohol (from F6)	(11 CRUD) (13 F21) (10 F19) (7 F6) (8 F6)
RC–CH–C(R)OH	F14	Tertiary alcohol (from F6)	(8 CRUD) (9 CRUD) (11 CRUD)
O–C–C–CHOH	F15	Secondary alcohol (from F5)	(4 F9) (7 F5) (8 F5) (13 F23) (10 F22)
AcOC–CH–CHOAc	F16	Diacetate of F9	(1 F9) (4 F9) (5 F9) (14 F9) (15 F9)
EEOC–CH–CHOEE	F17	Bis(ethoxyethyl ether) of F9	(2 F9) (9 F9) (8 CRUD)
RC–CH–CHOEE	F19	Ethoxyethyl ether of F13	(8 F6) (2 F13) (9 F13)
RC–CH–CHOAc	F21	Acetate of F13	(1 F13) (4 F13) (5 F13) (14 F13) (15 F13)
O–C–C–CHOEE	F22	Ethoxyethyl ether of F15	(8 F5) (2 F15) (9 F15)
O–C–C–CHOAc	F23	Acetate of F15	(4 F9) (1 F15) (5 F15) (6 F15) (14 F15) (15 F15)

[a] For each reagent–product pair, the reagent number is as defined in Table I with the addition of alkaline hydrogen peroxide (NaOOH, no. 16) and dialkyl cuprate (RCu, no. 17). For solution of expanded problems Table I was modified to include the action of reagents 16 and 17 on the functional groups therein (e.g., CHO —NAOOH→ COOH). [b] These functionalities break into eight interconvertible equivalence classes, [F1,F2,F3,F7,F8], [F6,F12,F13,F19,F21], [F5,F10,F15,F22,F23], [F9,F16,F17], [F11], [F4], [F14], and [CRUD].



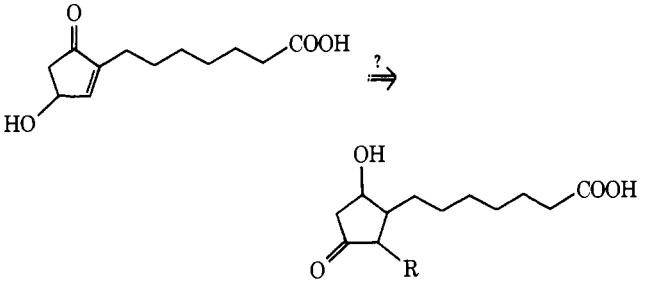
tion of the two tables, one may solve an expanded set of FGS problems. The problems may be expressed as

[C=C—CO RCHOH COOH]
⇒ [RC—CH—CO RCHOH COOH]

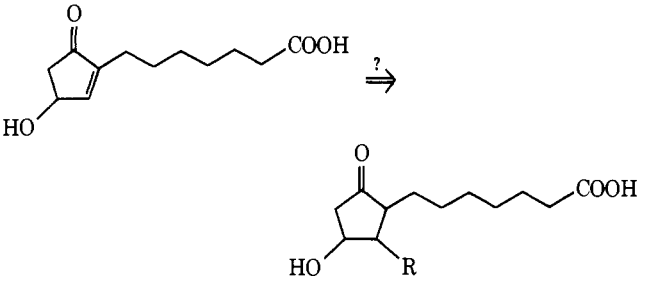
and

[C=C—CO RCHOH COOH]
⇒ [RC—CH—CHOH RCO COOH]

respectively with the solutions $\{CH_2N_2\ Ac_2O\ RCu\ OH^-\}$ for (3) and $\{CH_2N_2\ Ac_2O\ RCu\ NaBH_4\ EVE\ OH\ CrO_3PY\ H_3O\}$ for (4). Assuming that the three functionalities are separated, these two are shortest routes for the desired interconversions. One would like to use the expanded reaction dictionary to solve problems such as



and in fact the solution to (4) above accomplishes this. This is fortuitous, however, since the hydroxycyclopentenone is not an isolated enone and secondary alcohol as is seen on considering the consequences of application of the solution sequence for (3) to the concrete case.



Conclusions

Heuristic 3, in conjunction with a depthfirst search procedure, seems to suffice for solution of FGS problems of some complexity. Although analyzed in terms of the FGS structural model, this approach should be applicable to any structural model that allows one to express the search space as a reaction graph computable from some reaction dictionary and which contains recognizable subproblems that can be solved in an explicit and exhaustive manner. The limitations of this approach flow directly from the structural representation employed. Real molecules are not in fact (alas) ordered sets of isolated functional groups. Not all synthetic conversions can be represented as an FGS problem, and even when they can one has a nontrivial problem in translating a particular desired synthesis into an FGS problem format. Also this approach assumes a specified starting material whereas the conventional picture of synthesis has starting materials only analytically defined ("less than four carbons").[12]

References and Notes

(1) Approximately 12K of 36 bit memory is available in the PUFFT Fortran system. The compiler is described in "Programming Systems and Lan-

guages", S. Rosen, Ed., McGraw-Hill, New York, N.Y., 1967, pp 253–263.
(2) H. W. Whitlock, unpublished manuscript.
(3) For a discussion of decidability, see J. E. Hopcroft and J. D. Ullman, "Formal Languages and Their Relation to Automata", Addison-Wesley, Reading, Mass., 1969.
(4) A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The Design and Analysis of Computer Algorithms", Addison-Wesley, Reading, Mass., 1974, pp 173–223, 318–334.
(5) J. R. Slagle, "Artifical Intelligence: The Heuristic Programming Approach", McGraw-Hill, New York, N.Y., 1971.
(6) N. J. Nilsson, "Problem-Solving Methods in Artifical Intelligence", McGraw-Hill, New York, 1971, pp 43–79.
(7) J. Doran, "An Approach to Automatic Problem-Solving, in Machine Intelligence", Vol. 1, N. L. Collins and D. Michie, Ed., American Elsevier, New York, N.Y., 1967, pp 105–123; D. Michie, *ibid.*, pp 135–151.
(8) J. Doran, "New Developments of the Graph Traverser in Machine Intelligence", Vol. 2, American Elsevier, New York, N.Y., pp 119–135.
(9) E. Dijkstra, *Numerische Mat.*, **1**, 269 (1959), discussed in ref 6.
(10) Because of the assumption that all reactions that work do so equally well (all yields the same), any shortest path is considered sufficient. The set theoretic expansion of algorithm II to produce all shortest routes is straightforward.
(11) I. Pohl, "Bi-Directional Search, in Machine Intelligence", Vol. 4, B. Meltzer and D. Michie, Ed., American Elsevier, New York, N.Y., 1971, pp 127–140.
(12) I gratefully acknowledge partial support of this work by the National Science Foundation.

The Solvatochromic Comparison Method. 3. Hydrogen Bonding by Some 2-Nitroaniline Derivatives

Tai Yokoyama,[1a] R. W. Taft,*[1b] and Mortimer J. Kamlet*[1c]

Contribution from the Faculty of Literature and Science, Toyama University, Gofuku, Toyama, Japan, the Department of Chemistry, University of California, Irvine, California 92664, and the Naval Surface Weapons Center, White Oak Laboratory, Silver Spring, Maryland 20910. *Received September 8, 1975*

Abstract: Enhanced solvatochromic shifts in hydrogen-bond acceptor (HBA) solvents for 2-nitroaniline, 2-nitro-*p*-toluidine, and 2-nitro-*p*-anisidine relative to their *N,N*-dimethyl derivatives show good linear correlation with the β-scale of solvent HBA basicities. Reciprocally, the new experimental results are used to expand the data base which supports the β-scale. Solvatochromic comparison between *N*-methyl- and *N,N*-dimethyl-2-nitro-*p*-toluidine shows hydrogen bonding in the former compound to be *intra*molecular in all solvents studied.

In documenting solute-solvent hydrogen-bonding interactions by the solvatochromic comparison method,[2] three important conditions need to be met. (a) First, a plot of corresponding ν_{max} values (or other appropriate spectroscopic or free-energy properties) for two solutes of differing hydrogen-bonding ability in a series of solvents of varying polarity, but wherein hydrogen bonding is excluded, should show a linear relationship with a statistically acceptable correlation coefficient; this establishes how the spectra are influenced by changing solvent polarity. (b) Next, data points representing solvents in which hydrogen bonding occurs should be displaced from the regression line (all in the same direction) by statistically significant amounts; the deviations are presumed to reflect specific solute-solvent interaction effects. (c) Finally, the direction of the displacements should be consistent with the chemistry involved, and the relative magnitudes should reflect a reasonable order of solvent hydrogen-bond donor (HBD)[3] strengths in the case of solvent to solute (type A)[4] bonding or solvent hydrogen-bond acceptor (HBA) strengths where the effects derive from solute to solvent (type B)[4] hydrogen bonds.

In an earlier paper,[5] we employed the method to evaluate the spectral effects of type-B hydrogen bonding by 4-nitroaniline and 4-nitrophenol to a series of HBA solvents and to provide data toward the formulation of a β-scale of solvent hydrogen-bond acceptor basicities. We shall now use the solvatochromic comparison method and the β-scale to assess hydrogen bonding and solvent polarity effects on the electronic spectra of several 2-nitroaniline derivatives.

Hydrogen Bonding by 2-Nitroaniline. Values of ν_{max} in 25 solvents for the [$>^{+}N{=}C(1) \rightarrow C(2){=}NO_2^{-}$] electronic transitions of 2-nitroaniline (**1**) and *N,N*-dimethyl-2-nitroaniline (**2**) are assembled in Table I and plotted against one another in Figure 1. Compound **1**, but not **2**, can act as an HBD solute. The solvents are of three types: nine which are considered to be neither HBD's or HBA's (or such weak acceptors that their pK_{HB}'s would be anticipated to be lower than -0.5)[6] are represented by open circles in the figure; seven hydrogen-bond bases ($pK_{HB} > 0.7$)[6] are represented by filled circles; and nine amphiprotic R-OH solvents (capable of acting as HBD acids or HBA bases) are represented by triangles.

It is seen that the results fulfill the first requirement for solvatochromic comparison in that excellent linear regression is observed for the data in the nine non-hydrogen-bonding solvents. The least-squares correlation equation is

$$\nu(\mathbf{1})_{max} = 0.874\nu(\mathbf{2})_{max} + 4.51 \text{ kK} \quad (1)$$

with $n = 9$, r (the correlation coefficient) = 0.991, and SD (the standard deviation) = 0.08 kK [kK (kilokaysers) = $cm^{-1}/1000$].

Condition b (above) is also easily satisfied by the results in the HBA and amphiprotic solvents. Displacements from the regression line are all in the direction of lower transition energies for the HBD substrate **1** relative to the non-HBD substrate **2** and range from 2.1 to 12.6 SD's of eq 1. The enhanced bathochromic shifts attributable to hydrogen bonding by **1** to the HBA solvents, $-\Delta\Delta\nu(\mathbf{1}\text{–}\mathbf{2})^{B}_{\leftarrow H_2N}$,[7] calculated from

$$-\Delta\Delta\nu(\mathbf{1}\text{–}\mathbf{2})^{B}_{\leftarrow H_2N} = \nu(\mathbf{1})_{max}^{\text{calcd, eq 1}} - \nu(\mathbf{1})_{max}^{obsd} \quad (2)$$

are included in Table I.

An electronic transition from a ground state resembling **1a** to an excited state more like **1b** should lead to strengthening of the type B hydrogen bond in the electronic excitation, so that the bathochromic effect of association between **1** and HBA solvents is as anticipated. The ordering of the $-\Delta\Delta\nu$ values for the amphiprotic solvents, 2-methyl-2-propanol (solvent 101) > 2-propanol (102) > 1-butanol (103) > ethanol (104) > methanol (105) > water (111) confirms that in these solvents also we are dealing *primarily* with type B hydrogen bonding phenomenology, rather than a type A effect (like **1c**).[8]