THE MOLECULAR MATCHING PROBLEM

by

Professor Rex K. Kincaid
Department of Mathematics
The College of William and Mary
Williamsburg, VA 23187

Abstract

Molecular chemistry contains many difficult optimization problems that have begun to attract the attention of optimizers in the Operations Research community. Problems including protein folding, molecular conformation, molecular similarity, and molecular matching have been addressed. Minimum energy conformations for simple molecular structures such as water clusters, Lennard-Jones microclusters, and short polypeptides have dominated the literature to date. However, a variety of interesting problems exist and we focus here on a molecular structure matching (MSM) problem.

Consider two molecular structures, A and B, of similar size. We are asked to determine how similar the two structures are. One of the molecular structures, say A, and its inter-atomic distance matrix D_A is known apriori. The only information known for the second structure, B, is its inter-atomic distance matrix D_B. If the two molecular structures have the same number of atoms then the objective is to find a permutation (represented by a permutation matrix P) of the rows and columns of D_B, so that the absolute value of the entry-wise matrix difference, $D_A - PD_BP^T$, is as small as possible. In the case when there exists a permutation yielding a difference of zero, then the two molecules are identical (up to rotations and translations). The MSM problem was first studied by Barakat and Dean (1990a,b).

In the ensuing presentation we will focus solely on the structural matching problem when the two molecular structures have the same number of atoms. Similar techniques can be applied to the case when the molecular structures are unequal in size. In particular, we will study randomly generated structures generated in the unit cube and two DHFR protein structures (L. casei and E. coli) obtained from the Brookhaven protein data bank. First, we formulate the MSM problem as a 0/1 nonlinear integer programming problem and discuss the performance of a linearization technique due to Oral and Kettani (1992). Next, a tabu search heuristic is described and its performance summarized.

A 0/1 nonlinear integer program can be constructed for the MSM problem. The 0/1 decision variables p_{ij} are the entries in the permutation matrix P. In addition to the 0/1 restriction, each row and column of P must have exactly one entry with a value of 1. This is reflected in the following constraints.

$$\sum_{i=1}^{n} p_{ij} = 1 \text{ for all } j = 1..n \text{ and } \sum_{j=1}^{n} p_{ij} = 1 \text{ for all } i = 1..n. \tag{1}$$

The entries in the matrix D_A and D_B represent the euclidean distances between the atoms (or points in $\mathcal{R}^3$). As a result the distance matrices are symmetric with zero entries on the main diagonal. Two objective functions are considered;

$$(2)\ \min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |d_{ij}^A - \sum_{k=1}^{n} \sum_{l=1}^{n} p_{ik} d_{kl}^B p_{jl}| \text{ and } (3)\ \min \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (d_{ij}^A - \sum_{k=1}^{n} \sum_{l=1}^{n} p_{ik} d_{kl}^B p_{jl})^2.$$

Objective function (2) appears to be simpler, in that, when expanded, the resulting polynomial is of degree two. However, the absolute value terms are difficult to implement in most standard optimization packages. Objective function (3), when expanded, results in a fourth order polynomial. This polynomial can be dramatically simplified by imposing the 0/1 restrictions and the constraints (1). In fact, all third and fourth order terms vanish and we are left with a 2nd degree polynomial. Consequently we use (3) so that we can avoid the absolute value terms of (2). The resulting 0/1 quadratic integer program (QIP) will have n^2 decision variables and $2n$ linear constraints.

One approach to solve this 0/1 QIP (or at least provide an upper bound on the optimal objective function for the heuristic search) is to first linearize the quadratic objective function so that a 0/1 integer linear

program (ILP) results. By doing this we now have a variety of commercially available linear programming (LP) and branch and bound codes at our disposal. The LP codes can solve the 0/1 ILP if we relax the 0/1 restrictions. That is, if we replace $p_{ij} = 0, 1$ with $p_{ij} \geq 0$ and $p_{ij} \leq 1$. The hope is that the optimal solution to the LP relaxation will have many of the p_{ij} close to 0 or 1 and that this solution can be used as a *hot start* for the heuristic search procedures. If this does not happen the branch and bound codes will provide feasible solutions (and perhaps optimal solutions), but at a much greater computational effort.

Several linearizations procedures exist for quadratic 0/1 integer program with linear constraints. We have chosen the most compact formulation available (due to Oral and Kettani 1992) since we hope to be able to solve relatively large problems. Their linearization of our problem results in $2n^2$ decision variables and $n^2 + 2n$ regular constraints plus n^2 upper bound constraints of the form $p_{ij} \leq 1$ (non-negativity of the decision variables is assumed by all commercial codes).

The above techniques were tested on $n = 10$ and $n = 20$ size problems. The second being an 800 variable 440 constraint (with 400 upper bounds) size linear program. Both of the LP relaxations yielded optimal solutions with p_{ij} near $1/n$ for nearly all i and j. That is, almost none of the decision variables were close to 0 or 1. Consequently, this method did not yield an effective way to generate starting solutions for the heuristic algorithm. When $n = 10$ the branch and bound method of LINDO was used and a feasible solution to the 0/1 ILP (and 0/1 QIP) was found. It was not the optimal solution. When $n = 20$ LINDO was unable to solve the LP relaxation and the CPLEX code at Purdue University was used to determine the optimal solution. A branch and bound method has not been attempted for the $n = 20$ problem. Hence, it is still unknown whether the branch and bound method will yield good starting points for the heuristic algorithm.

Tabu Search (TS) incorporates conditions for strategically constraining and freeing the search process, as well as memory functions of varying time spans to intensify and diversify the search. The search proceeds from one solution to another via a move function and attempts to avoid entrapment in local optima by constructing a *tabu list* which is simply a list of previously selected moves. These previously selected moves are then deleted from the set of potential moves associated with the current solution. The underlying assumption is that prohibiting the repetition of moves will prevent the return to earlier local optima.

For our purposes, the heuristic embedded in the TS will be a local improvement scheme. Beginning with an initial feasible solution to an optimization problem, the local improvement procedure attempts to improve upon the trial solution by making small (incremental) changes. A rearrangement operation is applied until a perturbed configuration with an improved objective function value is discovered. The improved arrangement becomes the new trial solution, and the process is repeated until no further incremental improvements can be made. The basic solution approach for TS consists of a *construction* phase that generates a starting solution (hopefully from the 01 ILP) and an *improvement* phase that seeks to iteratively improve upon the starting solution. After max_it iterations of the improvement phase we do one of the following; *intensify* the search by restarting the improvement phase at the current best solution; or *diversify* the search by restarting the improvement phase in an unexplored region of the solution space; or stop and display the best solution found. For additional information about TS see Glover (1990), Kelly et al (1991), and Kincaid and Berger (1993).

Several TS codes were written in Fortran 77 by the author. Tables 1 and 2 summarize the TS results for randomly generated data when $n = 10$ and $n = 20$. Column 4 lists the number of tabu moves. Column 5 lists the number of these moves that yielded the best objective function value yet encountered in the search and whose tabu status was over-ridden. Column 6 is *yes* if the recency-based diversification scheme was needed, and *no* otherwise. Column 7 is *yes* if the frequency-based diversification scheme was needed, and *no* otherwise. The parameter *memsiz* refers to the number of moves kept on the tabu list, while *stop* is the number of attempts allowed to escape from a local optimum. The maximum number of neighborhood searches allowed is given by *maxit*. All computational times reported are for a 386-based microcomputer without a math-coprocessor running at 16 MHz. The two DHFR protein structures–L. casei and E. coli–have 159 and 162 atoms respectively. A 20 atom portion of the L. casei protein was used to generate a series of test problems and the resulting MSM was found to be much more difficult than the randomly generated ones.

References

Barakat, M.T. and P.M. Dean, "Molecular Structure Matching by Simulated Annealing. I. A Comparison Between Different Cooling Schedules," *J. of Computer-Aided Molecular Design*, **4**, 1990, 295-316.

_____, II. An Exploration of the Evolution of Configuration Landscape Problems," *J. of Computer-Aided Molecular Design*, **4**, 1990, 317-330.

Glover, F., "Tabu Search: A Tutorial," *Interfaces*, **20**, 1990, 74-94.

Kelly, J.P, M. Laguna, and F. Glover, "A Study of the Diversification Strategies for the Quadratic Assignment Problem," Working Paper, University of Colorado at Boulder, 1991.

Kincaid, R.K. and R.T. Berger, "The Damper Placement Problem on Space Truss Structures," accepted s.t. revisions *Location Science* 1993.

Oral, M. and O. Kettani, "Reformulating Nonlinear Combinatorial Optimization Problems for Higher Computational Efficiency," *European J. of Operational Research*, **58**, 1992, 236-249.

#	itrs.	time (m)	tabu	asp.	d1	d2
1	7	0.14	0	0	no	no
2	7	0.15	0	0	no	no
3	6	0.12	0	0	no	no
4	18	0.35	67	0	no	no
5	21	0.41	69	0	no	no
6	5	0.10	0	0	no	no
7	7	0.14	0	0	no	no
8	6	0.12	0	0	no	no
9	7	0.14	0	0	no	no
10	8	0.16	0	0	no	no

Table 1. $n = 10$ tabu search results.

#	itrs.	time (m)	tabu	asp.	d1	d2
1	20	1.54	9	0	no	no
2	21	1.61	18	2	no	no
3*	68	5.26	339	5	yes	no
4	15	1.15	6	0	no	no
5	118	9.25	959	13	yes	yes
6	36	2.78	96	7	no	no
7	16	1.23	2	0	no	no
8	17	1.30	3	0	no	no
9	27	2.08	8	3	no	no
10+	134	10.51	1052	23	yes	no

(*) $stop = n$ (+) $maxit = 100$

Table 2. $n = 20$ tabu search results: $memsiz = n$, $stop = 2n$, $maxit = 50$.