THE UNIVERSITY OF MICHIGAN
COMPUTING RESEARCH LABORATORY

REPRESENTATION-SELECTION FOR CONSTRAINT SATISFACTION PROBLEMS: A CASE STUDY USING n-QUEENS

Bernard A. Nadel
CRL-TR-5-86

1079 East Engineering Bldg.
Ann Arbor, MI 48109

THE UNIVERSITY OF MICHIGAN

COMPUTING RESEARCH LABORATORY*

REPRESENTATION-SELECTION FOR CONSTRAINT SATISFACTION PROBLEMS: A CASE STUDY USING n-QUEENS

Bernard A. Nadel

CRL-TR-5-86

MARCH 1986

Room 1079, East Engineering Building
Ann Arbor, Michigan 48109
USA
Tel: (313) 763-8000

*Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agency.

REPRESENTATION-SELECTION FOR CONSTRAINT SATISFACTION PROBLEMS: A CASE STUDY USING n-QUEENS[1]

Prof. Bernard A. Nadel[2]

Dept. Electrical Engineering and Computer Science

University of Michigan

Ann Arbor, MI 48109

March 1986

[1] This work was in part supported by the Computer Science Dept. of Rutgers University, New Brunswick, N.J., and in part by the Dept. Electrical Engineering and Computer Science of the University of Michigan, Ann Arbor, MI. This technical report therefore appears as part of the Technical Report series of both departments — DCS-TR-182 at Rutgers U. and CRL-TR-5-86 at U. Michigan.

[2] Previously: Bernard A. Nudel.

REPRESENTATION-SELECTION FOR CONSTRAINT SATISFACTION PROBLEM

A CASE STUDY USING n-QUEENS

ABSTRACT

The purpose of this paper is three-fold: (i) to emphasize that a given real-world problem can often be represented as a Constraint Satisfaction problem in a *variety* of ways, (ii) to show how our recently developed complexity results for Constraint Satisfaction problem-solving may be used to provide quantitative guidance in selecting between such alternative representations, and (iii) to provide a case study of a general new approach to the development of theory-based heuristics, and in particular of *instance-specific* theory-based heuristics. These three topics are discussed respectively in sections 3, 4 and 5 below.

1. INTRODUCTION

It is well known that one of the major opportunities, as well as bottlenecks, in improving prob n-
solving efficiency lies in the choice of an appropriate problem-solving representation. This issue has ıg
been of interest in Artificial Intelligence, as exemplified by the work of Amarel [1] and others [3], [6],]].
However, previous studies have considered the problem only *qualitatively*, by studying the various alte a-
tive representations that are possible. The present paper gives initial results on what appears to be ıe
first *quantitative* approach for *choosing between* problem representations.

Our results refer to the Constraint Satisfaction or Consistent Labeling Problem[1] CLP, which i lf
has received much attention in Artificial Intelligence [2], [5], [7], [8]. However, the general metho is
applicable to arbitrary problem-domains, and to types of problem-solving guidance other than st
representation selection. Basically, the approach is to derive analytic expressions capable of gi ıg=
problem-solving complexity as a function of whatever problem-solving parameter the user is intereste in
making choices over. One can employ such an expression as a guide to problem-solving decision-ma ıg
simply by making that choice which minimizes complexity as predicted by the expression. If the cho es
range over algorithms, this approach provides a theory-based *algorithm selection heuristic*. If the cho es
range over search-orderings then one obtains a theory-based *search-order selection heuristic*. These ro
uses of our complexity results have been studied in [14], [15] and [10].

We consider here for the first time a third type of decision-making guidance available from our c n-
plexity expressions: advice on choosing good problem-solving representations. That is, we use our res ts
to provide a theory-based *representation selection heuristic*. This third use depends heavily on ro
features of our analyses: (i) precision and (ii) generality. Without *precision* it would be impossible to c n-
pare *individual* alternative CLP instances that provide differing formulations of the given problen of
interest. Without *generality* it would not be possible to compare *all* the widely differing alterna re
representations in CLP that arise for a given problem.[2] We have attained precision by incorporatir a
new parameter into the analysis: constraint *satisfiability* or *looseness*. We have attained generality ıy
dropping the many unnatural restrictions that are usually assumed in analyses of CLP, such as the res c-
tion to binary constraints, domains of equal size, etc.[3]

Section 2 discusses the Consistent Labeling Problem. Section 3 shows how the well-known *n*-qu ıs
problem may be represented by a variety of different CLP instances. Section 4 shows how our comple ;y
expressions (due to their precision and generality) can accurately predict problem-solving complexit; of
these widely differing, individual CLP instances, and hence can provide an accurate, formal mean of

---

[1] We use *problem* to denote an actual problem-situation (i.e. one for which a specific answer can be given) and *Proble* to denote a class of *problems*. We also use the term *Problem instance* or simply *instance* for a problem. Analogously, the *Const nt Satisfaction Problem* or *CLP* is a class of instances, the generic one being denoted *clp*.

[2] For a given real-world problem, good representations may of course exist using frameworks other than CLP — say us a Theorem Proving formulation. Handling this would then require an even greater "generalization" of the problem class so that i e- comes the union of all target problem classes, with a separate analysis for each component class. However, one of the main poin of this paper is that CLP *alone* may very well provide a rich set of alternative representations, and their relative merit needs to. ıd can now, be understood.

[3] Of course this precision and generality are also advantageous in algorithm selection and search order selection. But the re absolutely *essential* for representation selection, since one is by definition comparing individual alternative instances which r e- over, are virtually always of widely differing type.

ranking alternative representations within CLP. Section 5 discusses these issues at a more general level.

2. CLP Instances

A CLP instance consists of *variables*, *values*, *constraints* and a *goal*. More specifically, there is a set $Z = \{ z_i \mid 1 \leq i \leq n \}$ of n variables z_i. Each variable may take on values from its finite domain $d_{z_i} = \{ z_{ij} \mid 1 \leq j \leq m_{z_i} \}$ of m_{z_i} values z_{ij}. There is a set $\mathbf{C} = \{ C_j \mid 1 \leq j \leq c \}$ of c constraints. Each constraint C_j specifies which value-tuples for the constraint argument variables "satisfy" the constraint, where each value for a variable is chosen only from that variable's domain. Thus each constraint C_j specifies some subset $T_j \subseteq D_j$ of satisfying value-tuples from $D_j = \bigtimes_{z_i \in Z_j} d_{z_i}$, the cartesian product of the domains of the constraint's argument variables. (We call T_j the *relation* induced by constraint C_j. Its size $S_j = \mid T_j \mid$ we call the constraint *looseness* or *satisfiability*. The *potential size* of a constraint C_j is the product $M_j = \prod_{z_i \in Z_j} m_{z_i}$ of the domain sizes of its arguments. $R_j = S_j / M_j$ is the *looseness* or *satisfiability ratio* of a constraint C_j, and lies between 0 and 1. The number of arguments of a constraint is $A_j = \mid Z_j \mid$, the constraint *arity*. Examples of these quantities are shown in table 1) The goal in solving a CLP instance is to find *all* ways of assigning values to the n problem variables from their respective domains so that each constraint is satisfied.

Note that the above formulation is very general. It allows CLP instances with differing numbers of candidate values for different variables. Constraints may each individually be over arbitrary argument sets $Z_j \subseteq Z$, and hence may certainly each individually have arbitrary arity $A_j \leq n$. Moreover, a given set of problem variables may very well have one constraint, many constraints or no constraints at all, over it.

For determining $N(clp)$, the *number of nodes generated* or $C(clp)$, the *number of constraint checks performed* in solving a given CLP instance clp by the algorithms of interest here, all that counts is the number n of variables, the number m_{z_i} of values in the domain of each variable z_i, and the argument set Z_j and relation T_j of each constraint C_j. We have obtained expectations of both $N(clp)$ and $C(clp)$ as a function of all these parameters except that instead of incorporating the full constraint-relation information T_j for each constraint, we use the corresponding satisfiabilities $S_j = \mid T_j \mid$. The equivalence-classes induced by our analysis parameters we call *small-classes*. They have empirically been found to be quite *homogeneous* in that most subsumed instances of a given small-class have similar complexities of solution. The important consequence is that small-class *expected-case* complexities (we take instances to arise uniformly in a small-class) can thus provide good approximations to the *exact-case* complexity of most individual instances in the corresponding small-class.

Expected complexities of CLP problem-solving, in terms of both nodes and checks, have been derived for the fully general case in [10]-[12] over small-classes[4] for three different CLP algorithms: gBT, gFC and gwFC, which are generalized versions respectively of the traditional Backtracking algorithm, and of the Forward Checking and the word-wise (or bit-parallel) Forward Checking algorithms appearing in [4]. Use of specialized versions of these results for both (virtually instance-specific) theory-based algorithm-selection and search-order selection has been considered in our earlier papers [14] and [15]. Here we consider their use for what appears to be the first example of theory-based problem-representation selection.

3. Alternative CLP Representations of n-Queens

The n-queens problem provides an intuitively clear example by which to demonstrate the range of CLP instances that may be used to represent a given real-world problem. We will talk of the q-queens problem to avoid confusion with n, the number of variables of a CLP instance. The q-queens problem is to find all ways to place q (indistinguishable) queens on a $q \times q$ chess board so that no two queens attack each other. Six alternative CLP formulations of this problem are presented below. Figure 1 presents these alternatives schematically for the case of the 4-queens problem, showing also the two solution vectors in each case.

Representation Q1 (Row-Based): If there is any solution to the q-queens problem, a little thought shows that it must place exactly one queen in each row of the $q \times q$ chess board. It thus remains to find in which column the queen of each row is to be placed. The standard CLP representation of q-queens thus associates a CLP variable z_i with row i of the chess board, letting each variable take its values from

[4] As well as over *big-classes* — which are not directly relevant for our present purposes.

the same domain $d_{z_i} = \{ 1\ 2\ .\ .\ q \}$ where value j for variable z_i denotes that the queen of row i is in column j.

The interpretation here given to variables z_i itself ensures that no two queens are in the same row. For queens not to attack each other it remains then only to ensure that no two queens be in the same column or diagonal. This is expressed mathematically as

$$(z_i \neq z_j) \bigwedge (\mid z_i - z_j \mid \neq j - i) \qquad \forall\ i < j \in \{ 1\ 2\ .\ .\ q \} \tag{1}$$

For the specific case of 4-queens, the above constraints are shown in canonical form $C_j = (Z_j\ T_j)$ in table 1. The two solutions (consistent labelings) for 4-queens under this representation are (2 4 1 3) and (3 1 4 2). These are shown respectively by the positions of the Q's in the left and right top two boards of figure 1. To express the above, and later, constraints more transparently we introduce the four functions $r(z_i)$, $c(z_i)$, $rd(z_i)$ and $ld(z_i)$ giving respectively the row, column, right-diagonal and left-diagonal[5] corresponding to the value of variable z_i. In terms of these, constraints (1) become

$$c(z_i) \neq c(z_j) \bigwedge rd(z_i) \neq rd(z_j) \bigwedge ld(z_i) \neq ld(z_j) \qquad \forall\ i < j \in \{1\ 2\ .\ .\ q\} \tag{2}$$

Representation Q1′ (Column-Based): Note that in the CLP instance resulting from representation Q1 above, all variables z_i have the same domain d_{z_i} but no two variables may take on the same value. For such CLP instances a natural transformation would be to consider the domain values as variables and the original variables as the new domain values. In the q-queens case, this alternative formulation amounts to associating variables with board columns instead of with rows, and associating domain values with rows instead of with columns. Due to the symmetry of the q-queens situation this alternative version is isomorphic to representation Q1 above. However in general this will not be so, and the transformed CLP instance where variables become values and vice versa may provide an improved representation and should be considered.

Under this representation, with variables and values associated with columns and rows in the manner shown for Q1′ in figure 1, the two consistent labelings (2 4 1 3) and (3 1 4 2) of Q1 become respectively (3 1 4 2) and (2 4 1 3). The two solutions for Q1 have thus been mapped into each other, reflecting the fact that the two corresponding physical board configurations are related by a reflection in the main diagonal of the board.

Representation Q2 (Queen-Based): Rather than specifying a solution in terms of which column the queen of each row is in, we might just as well give it in terms of which square each of the queens is in. Therefore, associate a variable z_i with each of the q queens, and let each variable take a value indicating the square in which it will be placed. Labeling the squares in some way from 1 to q^2, the domains are thus $d_{z_i} = \{1\ 2\ .\ .\ q^2\}$, the same for each variable.

Table 1: Parameter values for the $c = \binom{4}{2} = 6$ constraints C_j of the Q1 representation of 4-queens.

j	Z_j	T_j	A_j	M_j	S_j	R_j
1	$\{z_1\ z_2\}$	{ (13) (14) (24) (31) (41) (42) }	2	16	6	6/16
2	$\{z_1\ z_3\}$	{ (12) (14) (21) (23) (32) (34) (41) (43) }	2	16	8	8/16
3	$\{z_1\ z_4\}$	{ (12) (13) (21) (23) (24) (31) (32) (34) (42) (43) }	2	16	10	10/16
4	$\{z_2\ z_3\}$	{ (13) (14) (24) (31) (41) (42) }	2	16	6	6/16
5	$\{z_2\ z_4\}$	{ (12) (14) (21) (23) (32) (34) (41) (43) }	2	16	8	8/16
6	$\{z_3\ z_4\}$	{ (13) (14) (24) (31) (41) (42) }	2	16	6	6/16

[5] A right-diagonal is one that falls as it is traversed from left to right. A left-diagonal rises from left to right.

Fig. 1: Schematic of six CLP representations, with corresponding solutions, for the 4-queens problem.

Q1: Row-Based

	1	2	3	4
z_1		Q		
z_2				Q
z_3	Q			
z_4			Q	

$d_{z_i} = \{1\ 2\ 3\ 4\}$

(2 4 1 3) (3 1 4 2)

Q1′: Column-Based

	z_1	z_2	z_3	z_4
1			Q	
2	Q			
3				Q
4		Q		

$d_{z_i} = \{1\ 2\ 3\ 4\}$

(3 1 4 2) (2 4 1 3)

Q2: Queen-Based

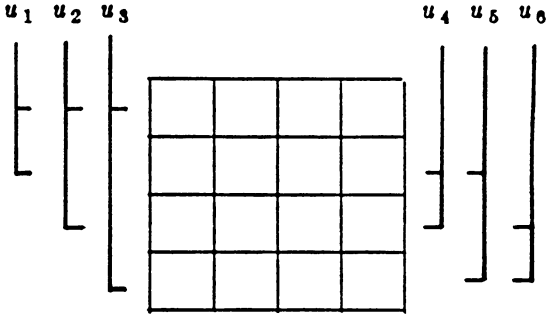
Q1 = z_1, Q2 = z_2, Q3 = z_3, Q4 = z_4

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

$d_{z_i} = \{1\ 2\ 3\ 4\ .\ .\ 15\ 16\}$

(2 8 9 15) (3 5 12 14)

Q3: Binarization of Q1



$d_{u_i} = T_i$ of table 1

((24)(21)(23)(41)(43)(13))
((31)(34)(32)(14)(12)(42))

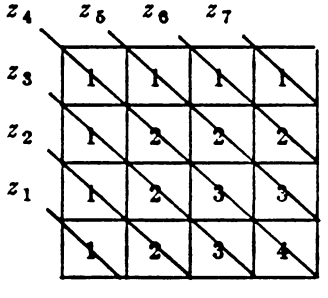
Q4: Square-Based

z_1	z_2	z_3	z_4
z_5	z_6	z_7	z_8
z_9	z_{10}	z_{11}	z_{12}
z_{13}	z_{14}	z_{15}	z_{16}

$d_{z_i} = \{0\ 1\}$

(0100000110000010) (0010100000010100)

Q5: Diagonal-Based



$d_{z_1} = d_{z_7} = \{0\ 1\}$
$d_{z_2} = d_{z_6} = \{0\ 1\ 2\}$
$d_{z_3} = d_{z_5} = \{0\ 1\ 2\ 3\}$
$d_{z_4} = \{0\ 1\ 2\ 3\ 4\}$

(0 1 3 0 1 2 0) (0 2 1 0 3 1 0)

Unlike under representation Q1, a queen may here a priori be on any square of the board. To ensure that no two queens attack each other we thus augment each of the $\binom{q}{2}$ constraints of (2) to include explicit avoidance of the same row, giving constraints

$$r(z_i) \neq r(z_j) \bigwedge c(z_i) \neq c(z_j) \bigwedge rd(z_i) \neq rd(z_j) \bigwedge ld(z_i) \neq ld(z_j) \quad (3)$$

for each combination of i and j from the integers 1 to q. Assuming the row-wise numbering of squares shown for Q2 in figure 1, the two solution tuples are (2 8 9 15) and (3 5 12 14). However, constraints (3) also allow as solutions any permutations of the above two solutions, such as (9 2 15 8) and (14 3 12 5). (In other words they do not correspond to indistinguishable queens as required.) One way to avoid permutations is to augment each constraint of the form in (3) with the extra requirement that lower-numbered queens be placed in lower-numbered squares; formally, one adds to (3) the additional conjunct $z_i < z_j$ (for $i < j$) to give $\binom{q}{2}$ binary constraints of the form

$$r(z_i) \neq r(z_j) \bigwedge c(z_i) \neq c(z_j) \bigwedge rd(z_i) \neq rd(z_j) \bigwedge ld(z_i) \neq ld(z_j) \bigwedge z_i < z_j$$

Representation Q3 (Group-of-rows-Based; Binarization of Q1): There is an important transformation applicable to any CLP instance, that we call *(arity) binarization* since the constraints of the resulting instance all have arity of two. This transformation is in fact implicit in Waltz' formulation of the line labeling problem in machine vision [16], in that he labels lines only indirectly by actually labeling junctions. Other examples of binarization appear in [10] in the context of solving Satisfiability problems and problems of join-formation in relational databases.

The idea behind the binarization transformation is a simple one. Instead of labeling the variables z_i of some CLP instance with candidate values from their associated domains d_{z_i} subject to appropriate (not necessarily binary) constraints, we label the constraint argument *sets* Z_j with candidate value *tuples* from the constraint relations T_j, subject to the *binary* constraints that each pair of argument sets must be labeled so that any common variable receives the same value.

Formally, this amounts to a transformation to a new CLP instance in which the constraint argument sets Z_j of the original instance become new CLP variables u_i. The old constraint relations T_j become the domains d_{u_i} of the new variables — which of course may require preprocessing to make the original T_j available in extensive form so that they can be used in instantiating the new variables u_i.[6] Note that the number of constraints c in the original instance equals the number of variables n in the new version and that the constraint satisfiabilities S_j of the original become the domain sizes m_{u_i} of the new version. For each pair of original argument sets Z_j that have one or more variables z_i in common, there is a binary constraint in the new instance between the corresponding new variables u_i. This constraint is simply that two value tuples (from the corresponding T_j) labeling the new u_i must assign the same values to any variables z_i common to the two u_i.

The Q3 section of figure 1 indicates the case for 4-queens where new variable u_i corresponds to argument set Z_i of table 1. Accordingly, the new domain d_{u_i} equals relation T_i of table 1. For example, since in the original instance the argument sets $Z_1 = \{z_1\ z_2\}$ and $Z_4 = \{z_2\ z_3\}$ have variable z_2 in common, there is a binary constraint in the new instance between the corresponding new variables u_1 and u_4. The constraint relation of this new binary constraint would be the following subset of $T_1 \times T_4$

{((13)(31)) ((14)(41)) ((14)(42)) ((24)(41)) ((24)(42)) ((31)(13)) ((31)(14)) ((41)(13)) ((41)(14)) ((42)(24))}

As required, the value of the common z_2 component of each pair of pairs is the same. Note that under this binarized version of representation Q1, the two 4-queens solutions become ((24)(21)(23)(41)(43)(13)) and ((31)(34)(32)(14)(12)(42)) where these are now value-tuples $(\bar{u}_1\ \bar{u}_2\ ..\ \bar{u}_6)$ whose component values are pairs of values of the original representation Q1. Binarization need not be applied only to representation Q1. New representations of q-queens may just as well be obtained by binarization of any initial representation, including the present already-binarized representation Q3.

Representation Q4 (Square-Based): A solution for q-queens can also be expressed in terms of whether or not a queen is present for each square of the board. We thus associate a variable z_i with each square

[6] On the other hand, a slightly different binarized version, that avoids the need for preprocessing, results by letting D_j, rather than their subsets T_j, be the domains in the new instance. The new binary constraints in this case would then need to incorporate an additional check that any value-tuple being assigned to a Z_j also satisfies the corresponding constraint C_j. This avoids the need for preprocessing to extract T_j, but incurs extra cost during problem-solving due to the more complex form of constraint checking.

$1 \leq i \leq q^2$, allowing each variable to take values from $d_{z_i} = \{0\ 1\}$, with 1 indicating the presence, and 0 the absence, of a queen on the corresponding square.[7] Under this representation the constraints that no two queens attack each other, become that no two squares on a common row, column or diagonal, may both contain a queen. We thus have the binary disjunctive constraints $(z_i = 0) \bigvee (z_j = 0)$ for each (unordered) pair of variables z_i, z_j that correspond to squares in the same row, column or diagonal. Note that for 4-queens there are 75 pairs of such squares — and hence 75 binary constraints of the above form. These constraints all have the same constraint relation $T_j = \{(0\ 0)\ (0\ 1)\ (1\ 0)\}$, and differ only in their arguments sets Z_j.

Under this representation, with variables associated with squares as shown for Q4 in figure 1, the two 4-queens solutions become (0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0) and (0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0). However, the constraints above ensure only that no two queens attack each other, and do not ensure that there are in fact q queens on the board. Thus other tuples (such as the tuple with all elements equal 0) besides the two given above will also result as solutions. One (weak) way to ensure that only solutions involving q-queens are found, is to incorporate the single additional q^2-ary constraint: $\sum_{1 \leq i \leq q^2} z_i = q$.

Representation Q5 (Diagonal-Based): We consider here our final example of an alternative CLP representation for the q-queens problem. It is analogous to representation Q1, but variables correspond to diagonals rather than rows. Representation Q1 resulted from the fact that a q-queens solution must have *exactly* one queen per row, and can thus be given in terms of the column position of the queen in each row. Analogously, q-queens solutions must have *at most* one queen per right diagonal and can thus be given in terms of the position of the queen within each diagonal, allowing however for each diagonal the possibility that no queen is present.

This suggests using a representation like that shown for Q5 in figure 1. A variable z_i is associated with each of the $2q - 1$ right diagonals, and each takes its values over the squares of the corresponding diagonal. An extra candidate value is included for each variable to allow for the absence of a queen in the corresponding diagonal. The resulting domains d_{z_i} are of course of unequal size. Using the convention (as shown for Q5 in figure 1) that integer i denotes the i-th square in a diagonal from the top left, while 0 denotes the absence of a queen, the two 4-queens solutions are then (0 1 3 0 1 2 0) and (0 2 1 0 3 1 0).

The interpretation here given to variables z_i itself ensures that no two queens are in the same right diagonal — in the same way that the interpretation given to the z_i in representation Q1 ensured that no two queens were in the same row. For queens not to attack each other, it remains then only to ensure that no two queens be in the same row, column or left diagonal. The necessary constraints under representation Q5 are therefore

$$\mathrm{r}(z_i) \neq \mathrm{r}(z_j) \ \bigwedge\ \mathrm{c}(z_i) \neq \mathrm{c}(z_j) \ \bigwedge\ \mathrm{ld}(z_i) \neq \mathrm{ld}(z_j) \qquad \forall\ i < j \in \{1\ 2\ .\ .\ 2q-1\}$$

Note there are now $\binom{2q-1}{2}$ binary constraints, rather than the $\binom{q}{2}$ in representations Q1 and Q2. But, analogously to the case for representation Q4, we also require here an extension of the above constraints to ensure that there are in fact q queens on the board. One possibility is to add the single constraint: $\sum_{1 \leq i \leq 2q-1} \delta(z_i) = q$, where $\delta(z_i)$ equals 1 if $z_i > 0$, and equals 0 otherwise.

4. Theory-Based Problem-Representation Selection

We have now presented five alternative CLP representations for q-queens problems. These differ in the number n of CLP variables involved, the set of domain sizes m_{z_i} for the variables, c the number of constraints, and the families of argument sets Z_j, arities A_j and constraint-satisfiabilities (or degrees of looseness) S_j of their constraints. All these factors have strong effects on problem-solving complexity. *All else being equal,* one can expect that a CLP instance is easier to solve if it has less variables, smaller domains for its variables, more and tighter constraints, and constraint argument sets such that constraint-checking can be done at earlier levels in the search tree.

Unfortunately, "all else is never equal" in practice. That is, it is usually the case that in comparing two alternative representations, each alternative has its own advantages in one or more of the above respects. How the representations compare in such a case is far from obvious. However, all these

[7] Note that if queens were to be considered distinguishable, we would allow variables to take values from $d_{z_i} = \{0\ 1\ 2\ .\ .\ q\}$, with 0 indicating no queen on a square, and the non-zero values indicating the different queens that may be on a square.

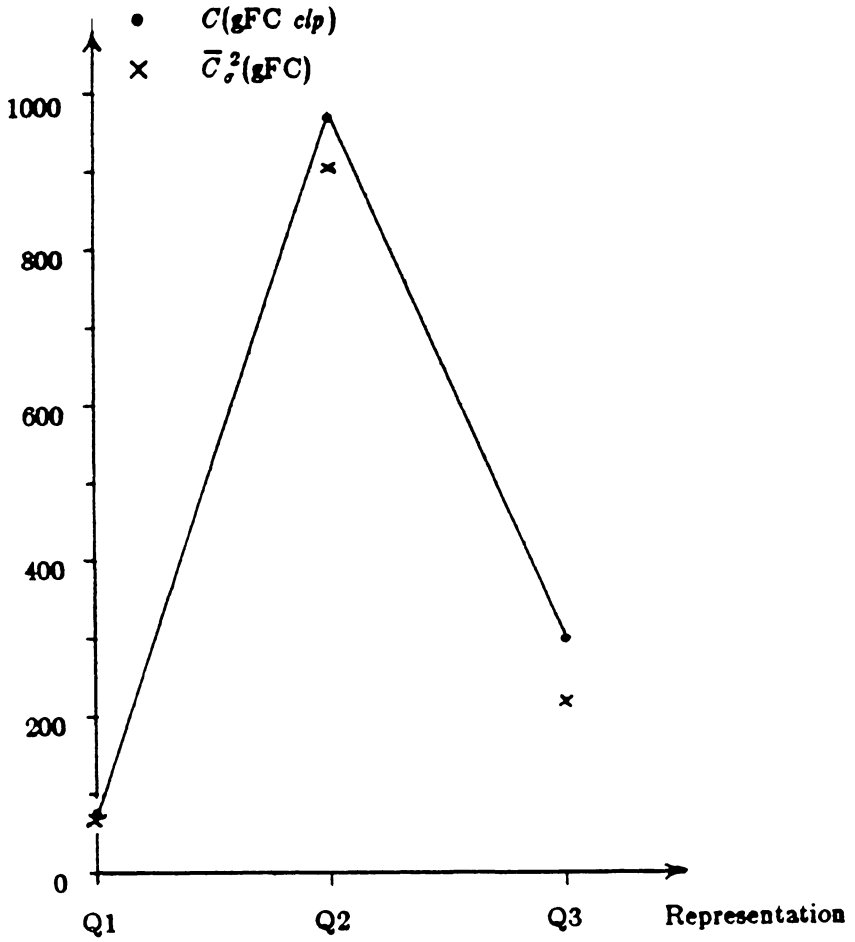
potentially conflicting factors have been incorporated into our complexity analyses, and the resulting complexity expressions provide a formal means of predicting how these influences combine in any given instance to effect its complexity of solution, and hence its complexity of solution relative to a competing alternative representation.

Figure 2 shows how this applies in choosing a representation for the 4-queens problem above. The figure shows for three of the representations, the theoretical and empirical complexity of solving the corresponding CLP instance using the gFC algorithm, where complexity is measured in terms of the number of constraint-checks performed. The empirical results in figure 2 are *exact-case* complexities found by solving the specific individual CLP instances concerned. The theoretical results are *expected-case* complexities from [10] or [12] for the small-class to which the corresponding instance belongs. (Note that analogous comparisons to those in figure 2 could have been made in terms of the number of nodes generated by gFC or for nodes or checks of the two other algorithms gBT and gwFC, since all corresponding complexity expressions are available in the above-mentioned papers.)

Of course, CLP problem-solving complexity is usually a strong function of the particular order in which variables are assigned values and of the order in which constraints are checked at each level of the search tree. In fact, as mentioned, since our analytic results capture these effects, another major use for them is in predicting good search orderings, as seen in [14] and [10]. For the particular experiments in figure 2, in the interests of saving space, the reader is referred to chapter 8 of [10] for details on the particular instantiation orderings and constraint-check orderings used.

In any case, we see from figure 2 that our analytic results are capable of providing accurate predictions of the complexity of solving the individual CLP instances that correspond to alternative formulations of 4-queens, and hence can be used to choose between these alternative representations. In particular (for the algorithm and search orderings used in the experiments) both theory and experiment agree that

Fig. 2: Empirical exact-case complexities C(gFC *clp*) and theoretical small-class expected complexities $\overline{C}_{\sigma}^{2}$(gFC) for three alternate CLP representations of 4-queens.



representation Q1 (the traditional one) is better than Q3, which is better than Q2. We have found similar agreement between theory and experiment in ranking problem representations for solving Satisfiability problems, Line-Labeling problems and Join-Formation problems in relational databases [10].

5. General Issues regarding Theory-Based Heuristics

This section summarizes and generalizes the above. At one level, this has been a discussion of representation-selection for n-queens. At a higher level it is a case-study of the ability of our analytic complexity expressions to provide a quantitative basis for selecting good problem-representations within CLP. And at a still higher level it exemplifies a new approach for the derivation of theory-based problem-solving heuristics. Specifically, the approach suggested here is to carry out a complexity analysis of problem-solving as a function of the decision parameter of interest and to use the derived expression to predict the complexity-minimizing decision. Theory-based guidance may thus be obtained for algorithm selection, search-order selection, and (given instance-specificity) for representation selection.

Several points deserve mention. First, it is important that the analysis be *precise* in the sense that predictions are accurate for individual instances. Most *expected-case*, *worst-case* or *best-case* complexity analyses do not have this instance-specificity, and are useful only with respect to whole sub-classes of problems (which are usually quite irrelevant in practice). But, as with our results, precision can be had even from one of the above three types of analysis, if the analysis parameters used induce sub-classes each of which is *homogeneous* in the complexity measure of interest. Then class-average results for example, can be used as good approximations for the *exact-case* complexity of most individual subsumed instances of the class. In our case, these homogeneous sub-classes are CLP *small-classes*, induced by employing in the analysis the new parameter *constraint satisfiability*. Analogous precision or instance-specificity can be achieved in other domains if a "homogenizing set" of parameters can be found in terms of which an analysis is tractable.

Secondly, *generality* of the assumed problem class is also important because theory-based guidance is then by definition available for a larger class of instances. But for meaningful theory-based representation selection, generality and instance-specificity are both *essential* since a single initial real-world problem corresponds to a set of individual CLP instances, usually of widely differing type, all of which must be compared. This has been one of the driving forces behind the successive generalizations introduced in our work from [13] to [14] to [15] to [10] and [12].

Thirdly, note that we have used our results here to provide theory-based guidance in the most straight-forward way: by comparing the computed complexities over individual competing (representation) decisions. An exhaustive approach like this is acceptable for representation selection and for algorithm selection, since there are usually only a handful of alternatives to compare in such cases. For other types of decisions, such as search-order selection, there are too many alternatives to allow explicit exhaustive comparison via the theory. In such cases *analytic minimization* (analogous to using calculus to find the condition under which a smooth function attains its minimum) rather than *exhaustive numerical minimization* becomes necessary. Analytic minimization was used in the formal derivation of search-order selection conditions for gFC in [14] and for gBT in [10]. As seen there, powerful new decision rules may result, or one may be able to rederive familiar old heuristics with the added benefit that the conditions under which they are valid is necessarily made explicit in the derivation. In [10] for instance, we show when it is valid to use such common Backtracking algorithm *search-ordering* heuristics as (i) instantiating variables in order of least number of candidate values, and (ii) checking constraints in order of decreasing tightness.

Finally, note that the potential for savings is greater than implied by the 4-queens example above. Figure 2 shows a complexity variation of an order of magnitude over the representations. In fact often variations of *many* orders of magnitude are possible as a result of varying the problem-representation, the algorithm or the search order used. The concomitant large savings possible by making an accurate complexity-minimizing choice show the importance of a formal, theory-based approach to decision-making such as used here rather than making problem-solving decisions based essentially on educated guess-work.

REFERENCES

[1] Amarel S., "On representation of problems of reasoning about actions," in *Machine Intelligence 3*, Michie D., Ed American Elsevier, New York, 1968.

[2] Gaschnig J., *Performance Measurement and Analysis of Certain Search Algorithms*, Dept. Computer Science, Carnegie-Mellon University, May 1979, Ph.D. dissertation.

[3] Gaschnig J., "A geometric model approach to representing graph-search problems: First results," *Proc. 3-rd Biennial Conf. Canadian Society for Computational Study of Intelligence*, Victoria, B.C., May 1980.

[4] Haralick R. M. and Elliot G. L., "Increasing tree search efficiency for constraint satisfaction problems," *Artificial Intelligence*, vol. 14, pp. 263-313, 1980.

[5] Knuth D. E., "Estimating the efficiency of Backtrack programs," *Mathematics of Computation*, vol. 29, no. 129, pp. 121-136, January, 1975.

[6] Korf R. E., "Towards a model of representation change," *Artificial Intelligence*, vol. 14, 1980.

[7] Mackworth A. K., "Consistency in networks of relations," *Artificial Intelligence*, vol. 8, pp. 99-118, 1977.

[8] Montanari U., "Networks of constraints: Fundamental properties and applications to picture processing," *Information Sciences*, vol. 7, pp. 95-132, 1974.

[9] Newell A., "On Representations of Problems," in *Annual Research Review* Dept. Computer Science, Carnegie Mellon U., pp. 19-33, 1966.

[10] Nadel B. A., *The Consistent Labeling Problem and its Algorithms: Towards Exact-Complexities and Theory-Based Heuristics*, Computer Science Dept., Rutgers University, New Brunswick, N.J., April 1986, Ph.D. dissertation.

[11] Nadel B. A., "The general Consistent Labeling (or Constraint Satisfaction) Problem," Report DCS-TR-170, Computer Science Dept., Rutgers University, New Brunswick, N.J., 1986, Also appears as Report CRL-TR-2-86, Dept. Electrical Engineering and Computer Science, U. of Michigan, Ann Arbor, MI, 1986. Submitted for publication in *J.ACM*.

[12] Nadel B. A., "Three consistent labeling algorithms and their complexities: search-order dependent and effectively instance-specific results," Report DCS-TR-171, Computer Science Dept., Rutgers University, New Brunswick, N.J., 1986, Also appears as Report CRL-TR-3-86, Dept. Electrical Engineering and Computer Science, U. of Michigan, Ann Arbor, MI, 1986. Submitted for publication in *J.ACM*.

[13] Nudel B. A., "Consistent Labeling Problems and their algorithms," *Proc Nat. Conf. on Artificial Intelligence (AAAI)*, pp. 128-132, Pittsburgh, August 1982.

[14] Nudel B. A., "Consistent-labeling problems and their algorithms: expected-complexities and theory based heuristics," *Artificial Intelligence (Special Issue on Search and Heuristics)*, vol. 21, no. 1 and 2, pp. 135-178, March 1983, Also in book: Search and Heuristics, North-Holland, Amsterdam 1983.

[15] Nudel B. A., "Solving the general consistent labeling (or constraint satisfaction) problem: Two algorithms and their expected complexities," *Proc Nat. Conf. on Artificial Intelligence (AAAI)*, pp. 292-296, Washigton D.C., August 1983, Also available as report DCS-TR-128, Computer Science Dept., Rutgers University, New Brunswick, N.J., 1983.

[16] Waltz D., "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, Winston P. H., Ed McGraw-Hill, New York, 1975.

AIIM SCANNER TEST CHART # 2

Spectra

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Times Roman

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Century Schoolbook Bold

4 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

6 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

8 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

10 PT **ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789**

News Gothic Bold Reversed

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Bodoni Italic

4 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

6 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

8 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

10 PT ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz;:'',./?$0123456789

Greek and Math Symbols

4 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

6 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

8 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

10 PT ΑΒΓΔΕΞΘΗΙΚΛΜΝΟΠΦΡΣΤΥΩΧΨΖαβγδεξθηικλμνοπφρστυωχψζ≧∓'',. / ≦±=≠°><⊁⊀⩾⩽≡

White

Black

Isolated Characters

e	m	1	2	3	a
4	5	6	7	o	°
8	9	0	h	I	B

A4 Page 6543210

MESH HALFTONE WEDGES

65

85

100

110

133

150

A4 Page 6543210

RIT ALPHANUMERIC RESOLUTION TEST OBJECT, RT-1-71

PRODUCED BY GRAPHIC ARTS RESEARCH CENTER

ROCHESTER INSTITUTE OF TECHNOLOGY, ONE LOMB

MEMORIAL DRIVE, ROCHESTER, NEW YORK 14623