Layout Planning as a Consistent Labeling Optimization Problem

D. Navinchandra David H. Marks

Intelligent Engineering Systems Laboratory
Department of Civil Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts, U.S.A.

ABSTRACT

Over the past decade the consistent labeling problem has received considerable interest in Artificial Intelligence. In particular, machine vision, cryptography and theorem proving. Many algorithms have been developed for solving consistent labeling problems. A majority of these algorithms are based on heuristic search. Techniques such as constraint propagation and forward checking have been used to reduce search effort.

Typically, the *consistent labeling problem* is formulated thus: there is a set of discrete variables, each with a finite set of values from which one is to be chosen. The task is to find labelings for all the variables such that a given set of constraints are simultaneously satisfied. The consistent labeling problem places few requirements on the nature of the variables or the constraints: the constraints can be either symbolic or numeric, monotonic or non-monotonic, continuous or discontinuous, linear or non-linear.

In this paper we propose a heuristic search technique for handling consistent labeling problems with the added requirement of generating a consistent labeling that optimizes a set of objectives. This new type of formulation is applicable to a wide variety of engineering problems. In this paper we have chosen layout planning as our example domain. The layout problem is formulated thus: there is a set of objects that have to be placed in a designated area where, each object has a finite set of locations (within the designated area) that it can be assigned to. The task is to find unique locations for all the objects such that the given set of constraints and objectives are simultaneously satisfied.

1. Introduction

A layout problem involves the placing of objects on a two dimensional space such that a set of inter-object relationships are satisfied. For example, in the Landscape design domain, a planner may deal with objects such as houses, parks, shopping-malls etc. While laying out these objects, the planner has to deal with constraints and objectives concerning soil conditions, visibility, noise, inter-object distance, access etc.

A simple landscape design problem can be formulated thus:

1. There is a set of landuses that have to be sited.
2. Each landuse is to be assigned to a lot of land, chosen from a given set of alternatives.
3. There is a set of constraints governing the labelings.
4. The problem is to find few (or all) ways of assigning landuses to lots such that the constraints are all simultaneously satisfied.

In the above formulation landuses are treated as variables and the lots are treated as values that can be assigned to variables. The act of assigning a value to a variable is known as "labeling" the variable.

In Artificial Intelligence (AI) literature, the above formulation is known as the Consistent Labeling Problem (CLP).

In general, a CLP is formulated thus: (Adapted from (Nadel 85a))

1. There is a set of variables.
2. Each variable has associated with it, a finite set of values. These sets are known as the domains of the variables.
3. There is a set of constraints on the values that various combinations of variables may compatibly take on, and
4. The goal is to find a few (or all) ways to assign to each variable, a value from its associated domain in such a way that all constraints are simultaneously satisfied.

In this paper, we propose a heuristic search technique for handling consistent labeling problems with the added requirement of generating a consistent labeling that optimizes a set of objectives. This new type of formulation is applicable to a wide variety of engineering problems. We call this problem the Consistent Labeling Optimization Problem (CLOP) and the corresponding Solution technique is called CLOPS.

We have chosen layout planning as our example domain. The layout problem is formulated thus: there is a set of objects that have to be placed in a designated area where, each object has a finite set of locations (within the designated area) that it can be assigned to. The task is to find unique locations for all the objects such that the given set of constraints and objectives are simultaneously satisfied.

The next section provides a scenario which shows the use of optimization in a layout planning problem. It shows how constraint relaxation could be used in dealing with overconstrained problems. Section 3 presents the major principles behind the CLOPS methodology. Section 4 presents the methodology with the aid of a representative example.

1.1. Background

The Consistent Labeling Problem (CLP) has received much attention in Artificial Intelligence and Operations Research. Over the years, the CLP formalism has been successfully applied to a wide variety of problems.

One of the important applications of consistent labeling is in machine vision. A typical problem in machine vision is scene analysis. The computer is given an image of a scene that is comprised of several regular shaped blocks. The computer is supposed to find out how many blocks are in the scene, even if the blocks partially obscure one another. Scene analysis involves three major steps: line/edge detection, line labeling and object interpretation. All three steps have been formulated as CLPs.

Another interesting application is to relational database retrieval. Some relational calculus operators, such as **join**, can be formulated as a CLP. Consistency-maintenance, redundancy-checking and query-answering in databases have also been treated as CLPs (Grossman, 1976)

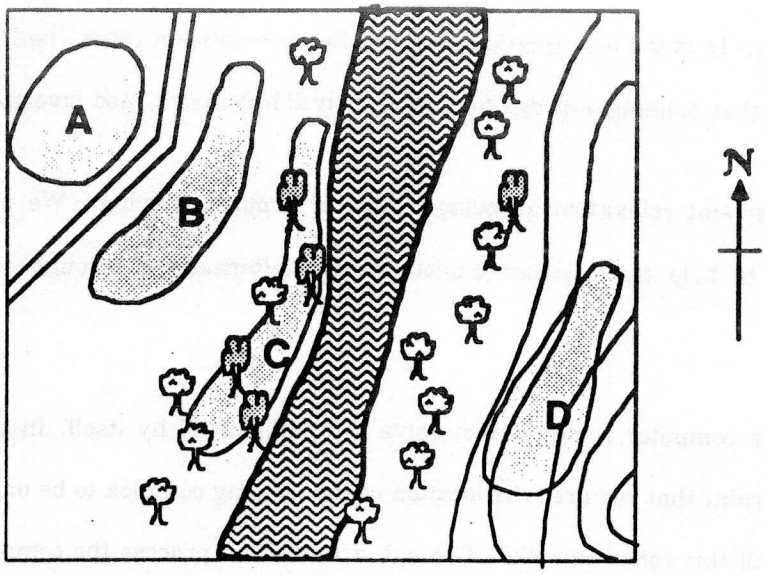
For a complete list of applications of CLP, refer the work of Nadel (Nadel 1985a, 1985b, 1985c, 1985d).

2. A Scenario

In this paper, we view layout problems as a CLOP. One of the important aspects of the CLOPS methodology is its ability to work through a staged search process while maintaining optimality over a set of objectives. The program is particularly good for problems that are overconstrained and can be solved only through the relaxation of constraints and the making of trade-offs among objectives.

Problems in the layout planning domain are often overconstrained and are amenable to the CLOP formulation. A layout designer who has many objectives to maximize is often faced with the problem of conflicting objectives. The designer goes through the process of recognizing these conflicts, describing them and then making tough decisions which involve tradeoffs. The proposed CLOPS methodology helps the designer go through this process.

The following scenario illustrates the importance of constraint relaxation in layout problems. The example is from the domain of landscape design.



The figure above shows a landscape on which a housing complex is to be sited. Let us consider how a landscape designer might approach this problem:-

The landscape designer starts the process by first identifying possible housing sites on the landscape. She then characterizes the sites as shown below:

- A is close to the road.
- B is close to the road like A, but also has easy access to the woods surrounding the river. B is a more scenic location and hence dominates A.
- C is a beautiful location as it is right up near the water and allows a beautiful view. This location will yield good profits from the sale of houses. However, she intends to use the river front area as a large picnic stop (an additional steady revenue). Site C conflicts with the development of such a picnic area.
- D is on a hill side and commands a very good view of the river and the surrounding valley. However, the designer decides not to place any development on the east side of the river. This is because it requires building a bridge which would surely overshoot the current budget.

Location A is easily eliminated, the real choice is between B and C. A decision has to be made. She can either trade beauty for dollars or look for ways around the problem. Through an exploratory process, she may realize that placing new development on D will both preserve the the river bank **and** provide a good vista. But this violates the constraint she set up earlier. She thinks

a while and finally decides on D as the best location. She justifies her decision thus: The D location allows profits to be so high that building a bridge to cross the river becomes a good investment.

This is how we see constraint relaxation as being a part of problem solving. We propose a computational framework to help the designer explore his/her alternatives through constraint relaxation.

It may be noted that the computer really cannot solve the problem all by itself. In the above example there was a constraint that required the location of the housing complex to be on the west side of the river. Let us call this constraint #25. Through a relaxation process the computer may make the suggestion about building on the other side of the river. The attitude the computer might take is that of: *Did you know that we could make a lot of money, provided you find some way of eliminating constraint #25?* It is up to the designer to realize that a bridge can be built. The computer's exploratory approach helps the designer be innovative. It is the computer's job to search the space making relaxations and presenting the user with some of the more promising resultant designs. In order to do this, the computer has to have some sense of optimality and it should know how to relax the constraints. These issues are discussed later in the paper.

3. Layout as a Consistent Labeling Optimization Problem

3.1. CLOPS: the formulation

We formulated our problem based on a formalism popularized by researchers in machine vision. The formalism is the Consistent Labeling Problem (CLP). Formally, the problem can be stated thus:

There is a set of variables **X** = { x1, x2, ... xn } Each variable has a set of discrete values associated with it. This set is called the domain of the variable:

$$D_{x,i} = \{ x_{i1}, x_{x2}, ... x_{im} \}$$

The problem is to select, for each variable, a value from its domain. This selection, is to be done while adhering to a set of constraints C.

$C = (c1, c2, \ldots cn)$

Where each constraint defines a condition on a finite set of variables:

$Ci = Pi(x_p, x_{p+1} \ldots)$

Where Pi is some predicate. The predicate may be numerical, non-linear, logical, boolean or of any reasonable form the user pleases.

The CLP has been addressed by several researchers. Some of the significant works are those of Mackworth (1977), Nadel (1985a, 1985b, 1985c, 1985d), Montanari (1974), Waltz (1974) and Freuder (1978). We propose to apply this rather generalized framework to the problem of landscape architecture.

The CLOPS methodology introduces the notion of objectives to the classical Consistent labeling problem stated above. The set of objectives **O** can be denoted as:

$O = (O1, O2, O3 \ldots)$

Wherein, each objective has to be either maximized or minimized.

Each objective function has a set of variables it depends on. This set is called its range and is denoted by **R**.

$R_{oj} = (x_p , x_{p+1} , \ldots x_r)$

Each $Xi \epsilon R_{oj}$ has a domain of its own $D_{x,i}$. For each $Xi \epsilon R_{oj}$ we can choose a value for Xi from its domain in order to calculate a value of Oj.

3.2. Use of Pareto Optimality

Pareto optimality is a powerful method for representing and understanding tradeoff situations. We shall now cast CLOPS in the light of Pareto Optimality, this will provide us with a graphical component.

Before plunging forward, let us examine how CLOPS handles ordinality:

In the CLOPS methodology, ranks are used to introduce ordinality. This is done by first computing the maximum & minimum of each objective taken individually. This can be shown on a number line:

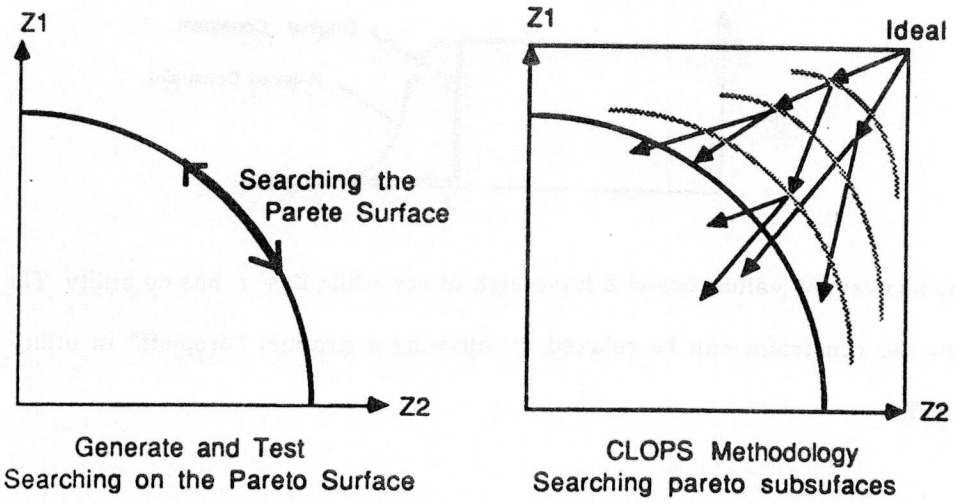
In the figure above, Z^* is the maximum and Z_* is the minimum value of some objective. The ranks are assigned starting from Z^*. We keep assigning ranks until we reach some lowest acceptable level Z_A. The rest of the values below Z_A are ranked as N where N indicates some very large number. It is assumed that the DM is indifferent among the values that fall within a particular rank.

The ranks obtained by a particular solution with respect to all the objectives is used to determine the optimality of the solution. All solutions that have a better rank (on at least one of the constraints and objectives) than all the other solutions are said to be Pareto Optimal.

Pareto Optimality, as defined above, is used to guide the search for solutions. At each stage of the search, the pareto set is found and expanded further. This process is repeated till a complete, pareto-optimal labeling is found. The node selection method is a cross between the best-first and beam-search strategies.

Traditional methods for dealing with multiple criteria problems have relied on a generate and

test paradigm. Under this paradigm, whole solutions were generated and tested for optimality. In the CLOPS methodology, partial solutions at different stages of the search are checked for optimality. This helps eliminate solutions even before they are fully instantiated, which in effect, drastically reduces the size of the search tree. These two methodologies are illustrated in the figures below:-



The generate and test technique tests complete solutions while, the CLOPS methodology starts its search from the ideal and slowly works its way down to the actual pareto surface.

3.3. Constraints & relaxation techniques

Each constraint has a set of variables it constrains. This set is called its range and is denoted by **R**. The arity of a constraint is equal to the size of the range **R**.

Traditionally constraints have been viewed as something that is either satisfied or not satisfied. Relaxing this black-n-white notion of constraint satisfaction, we introduce some gray/fuzzy areas into the constraint. Let us consider some examples:

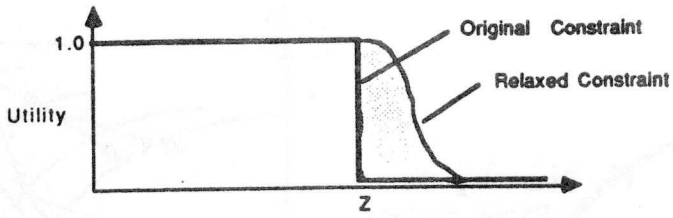
Example

Consider the constraint that:

Distance between the shopping complex and the parking lot should be less than 100 meters

This can be written as: $f(x) <= Z$

This can be plotted as:



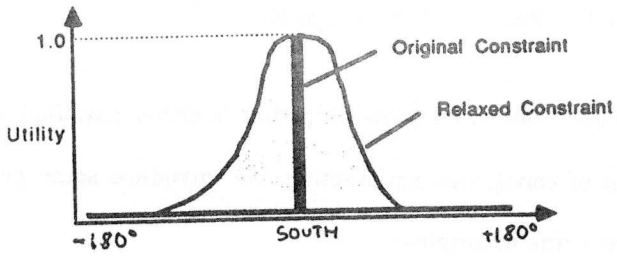
The solid line shows how values below **Z** have high utility while $Z + \epsilon$ has no utility. The dotted line shows how the constraint can be relaxed by allowing a gradual "drop-off" in utility rather than a sharp cliff.

Example

Consider the constraint that

The house shall face a direction = SOUTH

The corresponding plot is:

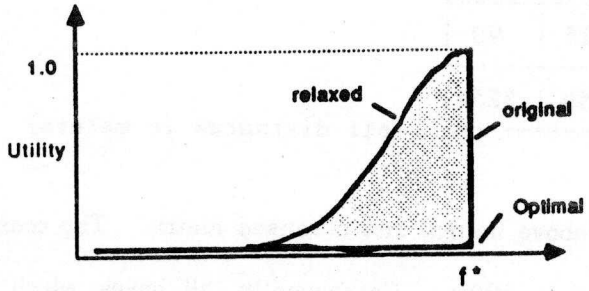


Example

Finally, consider an objective function that states:

MAXIMIZE: Distance between dumpsite and the housing complex

The corresponding plot is:



Where f^* is the optimal.

3.4. Computer Representation of Constraints and Objectives

All constraints and objectives are represented as matrices. For example the constraint:

Distance between the shopping complex and the parking lot should be less than 100m

is converted into a ranked matrix relating the possible sitings of the two landuses involved.

Let us assume that the possible sites for the shopping complex are: {a b c d} and the possible sites for the parking lot are: { p q r}.

In order to prepare a constraint or an objective for use by the CLOPS methodology , it is converted into a matrix by following a two step process: The first step is to set up a matrix indicating the distances between the alternate sites for the two landuses. The table below shows the distances between the sites (arbitrary numbers):

		Shopping Complex			
		a	b	c	d
Parking lot	q	150	80	90	132
	q	100	120	115	90
	r	300	312	150	325

(all distances in meters)

The second step is to convert the above matrix into a ranked matrix. The constraint given requires that the distance be less than 100m. Consequently, all boxes which satisfy this requirement are assigned ranks of unity. Boxes with values a little over 100m are assigned a rank of two and boxes with even higher values are assigned higher ranks. Boxes with unacceptably high distances are ranked with the letter N. The table below shows one possible assignment of ranks to combinations in the matrix:

		Shopping Complex			
		a	b	c	d
Parking lot	q	4	1	1	3
	q	1	2	2	1
	r	N	N	4	N

Both Constraints and Objectives are expressed in this fashion. Notice that the form of the original criterion is unimportant. Any non-monotonic, non-continuous, non-numeric criterion can be expressed as a matrix of arbitrary dimensions.

4. CLOPS: the methodology

We shall now present the CLOPS methodology, followed by an example.

4.1. The methodology

CLOPS is based on a heuristic search technique. The search tree is developed by sequentially choosing variables and branching on the variables' domains. This process alone will guarantee the generation of all possible solutions. It however, leads to a combinatorial explosion. To avoid this problem, nodes in the tree are expanded selectively. In order to select a node in the search tree we have adopted a best first strategy. This involves the assignment of a value to each node. The selection of a good node evaluator is important.

In our scheme, each node on the tree has an associated data set called its spectrum. The spectrum is simply a list of the ranks attained by all the objective functions. Hence, for O = (o1, o2, ... oj), a typical spectrum at node-n will be Sn = (Sn1, Sn2, ... Snj) The ideal solution is one, wherein all the ranks in the spectrum are known and are all equal to one. The CLOPS methodology, in addition to pareto optimality, uses a measure of deviation from the ideal spectrum. This deviation Δ is given by:

$$\Delta = \sum_{i=1}^{i=j} (Sni - 1)$$

Note that the objective functions are all weighted equally.

During the search process we have to keep evaluating partial labelings from one stage of the search to another. If a labeling is complete over all variables, its rank can be found by simply picking up the value from the appropriate box of a ranked matrix. On the other hand, it is not easy to find the ranks of a partial labeling. This is because, a partial labeling typically does not have enough information to access a ranked matrix with. For this reason, the ranks of partial labelings have to be estimated by using a heuristic. This concept is illustrated in the example below:

Consider a particular partial labeling which has only two variables instantiated: x1 and x2 with values "2" and "4" respectively. In addition, assume there are two criteria which have the

following ranked matrices:

Matrix 1:

x1 \ x2	2	3	4
1	N	1	2
2	N	1	2
3	N	1	2

Matrix 2:

x8 \ x2	2	3	4
1	2	1	2
8	2	N	2
7	3	1	1

The rank of the partial labeling with respect to Matrix1 is easy to find. We know the values of x1 and x2 and the rank is clearly equal to 2. Finding the rank of Matrix2 is not so easy. The second matrix requires that we know the value of variable x8. The rank can only be estimated. Since the value of one of the variables in Matrix2 is already known to be equal to "4" the rank has to be estimated using this known value. It can be seen that worst possible rank, regardless of what value x8 takes, is a 2 and the best possible is a 1 . In the CLOPS algorithm the best possible rank is taken as an estimate of the actual rank. This is done in order to guarantee optimality. For an explanation of why this heuristic guarantee's optimality, please refer Nillson's work on A^* search (Nillson '80).

4.2. A small example

Let us now look at a small representative example. Assume that there are three variables X1, X2 and X3. The domains of these three variables are *(a, b, c), (p, q, r) and (x, y, z)* respectively. The variables can be landuses such as schools or housing or a commercial area. There are two constraints and two objectives on the values of these variables.

The first constraint is called OP3. It concerns the viewshed for variable X1.

The matrix representation is:

OP3: Viewshed

X1:	a	b	c
Symbol	poor	good	moderate
Rank	4	1	2

The next matrix is OP4. It concerns the cost of variable X2.

OP4: Cost of X2

X2:	p	q	r
cost:	low	medium	high
rank:	1	2	3

There are two objectives. The first concerns the distance between X1 and X2. The second concerns the inter-visibility of X1 from X2. Both these objectives are simply stated as ranked matrices: OP1 and OP2 respectively.

OP1 : distance

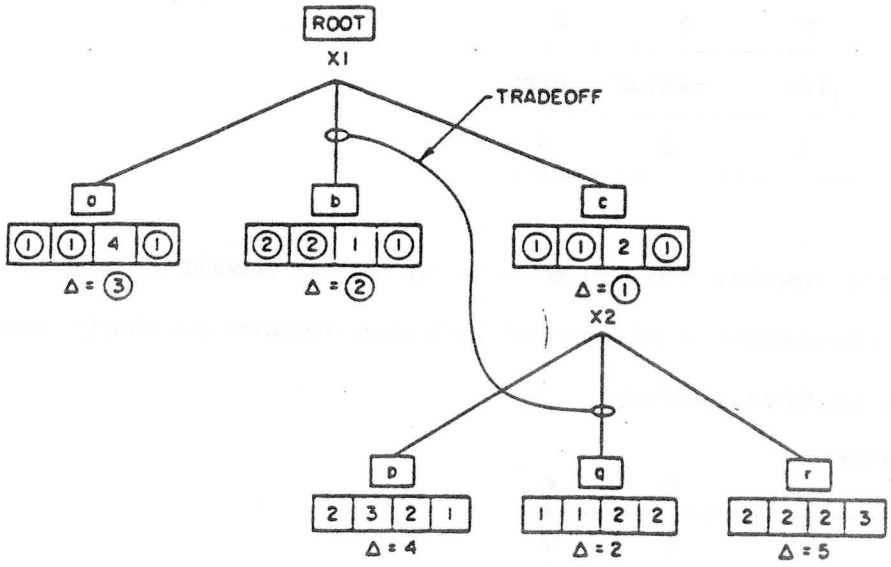
	p	q	r
a	1	2	2
b	2	2	2
c	2	1	2

OP2: Intervisibility

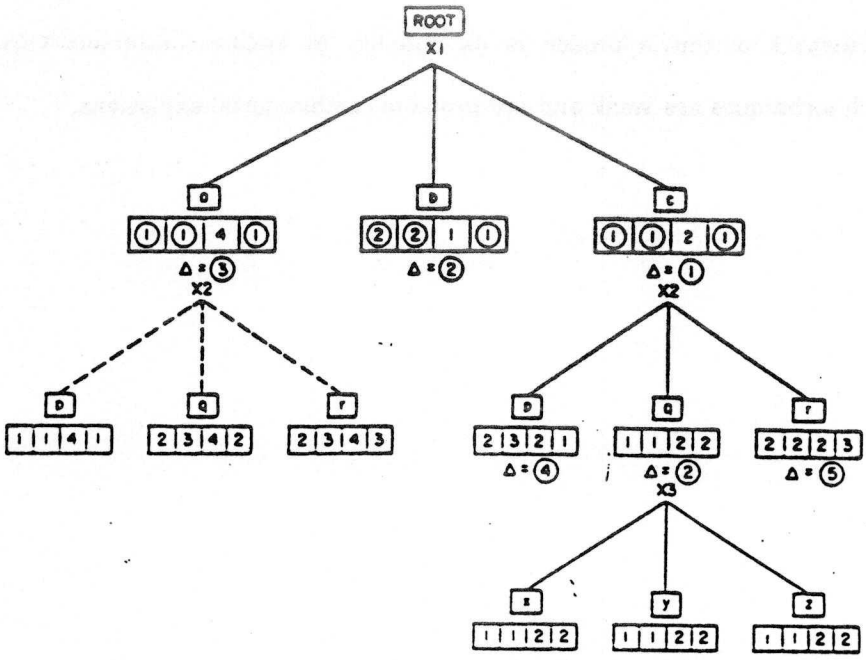
	p	q	r
a	1	3	3
b	4	4	2
c	3	1	2

Let us assume that the search starts by instantiating the variables in the order X1, X2, X3, X4. There is no particular reason for this, it is a random choice in this case. In the real program there are some heuristics used to determine the order of instantiation.

The search starts as shown in the figure below. The first variable X1 is instantiated into *a, b and c.* Under each choice, there is a bar of boxes. Each bar corresponds to the "spectrum" at that node. The values in the spectrum are the ranks attained by the different constraints and objectives. The values are in the order OP1, OP2 OP3 and OP4. The values that are circled, are the ones that are estimates of the true value. For example, under node "a" we know that the rank of OP3 is four. The rank of OP1 has to be estimated, the basis that X1 is already equal to "a", the best one could get from OP1 is a rank of one (i.e. for any value of X2).



Under each spectrum there is a Δ value. Using a best first search strategy, only the nodes with lowest Δ are chosen for expansion. Just after X1 has been expanded the best choice is node "c". It may be noted that choosing node "c" amounts to accepting a relaxation on constraint OP3. This is because it is of rank two. Later X2 is expanded as shown above. At this stage we have two nodes with Δ equal to two. We cannot decide which one to expand. These two nodes are "b" and "q". This indecision constitutes a tradeoff situation. The user is asked whether he/she is comfortable with low ranks on the distance and intervisibility objectives or low ranks on the constraints. Let us assume the user chooses the latter. The tree develops as shown in the figure below (ignore the dotted lines for now):



The user will most probably be content with node "q" and all its instantiations. However, we have an interesting quirk here. Notice that at Δ equal to three, the node "p" (dotted lines) is quite good. It gives us good ranks on both the objectives and on the cost constraint. It may be possible that the user may be interested in relaxing the viewshed constraint drastically in order to obtain a high rank on the cost constraint. Using a best first strategy alone, a user would have overlooked this choice. The CLOPS program searches for and finds such interesting constraint relaxations and shows them to the user. This is how exploration in the search tree can occur.

5. Conclusions

The CLOPS algorithm has been developed and tested in a landscape design system. The program CYCLOPS (Navinchandra '87) is built in LISP and runs on UNIX[1] based machines.

The CLOPS algorithm is applicable to a large number of engineering problems that involve a mix of constraints and objectives that may be non-linear, non-monotonic and even non-continuous.

---

[1]UNIX is a trademark of AT&T Bell Labs.

A major drawback of this approach is its inability to handle continuous variables. This is because search techniques are weak and are prone to combinatorial explosions.

REFERENCES & BIBLIOGRAPHY

Deo, Narshingh (1984) "Graph Theory" Prentice Hall 1984.

Fikes, R.E. (1970) "REF-ARF: A system for solving problems stated as procedures." AI Journal 1:27-120"

Freuder, E. C. (1978) "Synthesizing constraint expressions" Comm. ACM, 21 958-966.

Gross, Mark (1985) "Design by Constraints". Doctoral Thesis, Department of Architecture MIT.

Mackworth, A. K., (1977) "Consistency in networks of relations," Artificial Intelligence, vol, 8, pp. 99-118, 1977.

Montanari, U. (1974). "Networks of constraints fundamental properties and applications to picture processing". Information Sciences. 7 (1974) 95-132.

Nadel, B. A. (1985a). "The Consistent Labeling Problem, Part1: Background and Problem Formulation", Technical report, Dept. of Electrical Engg. & Computer Sci., University of Michigan, Ann Arbor. DCS-TR-164.

Nadel, B. A. (1985b). "The Consistent Labeling Problem, Part2: Subproblems, enumerations and constraint satisfiability" Technical report, Dept. of Electrical Engg. & Computer Sci., University of Michigan, Ann Arbor. DCS-TR-165.

Nadel, B. A. (1985c). "The Consistent Labeling Problem, Part3: The Generalized Backtrack Algorithm." Technical report, Dept. of Electrical Engg. & Computer Sci., University of Michigan, Ann Arbor. DCS-TR-166.

Nadel, B. A. (1985d). "The Consistent Labeling Problem, Part4: Forward checking and Word-Wise Forward Checking Algorithms", Technical report, Dept. of Electrical Engg. & Computer Sci., University of Michigan, Ann Arbor. DCS-TR-167.

D. Navinchandra, (1987) "Exploring for Innovative Designs by Relaxing Criteria and Reasoning from Precedents", Sc.D. Thesis, Intelligent Engineering Systems Lab., Department of Civil Engineering, MIT.

D. Navinchandra; Robert Lacey, (1985) "Steps towards automating Landuse planning guidelines" Internal Report, CERL 1985.

Nilsson, N.J. (1980) "Principles of Artificial Intelligence", Palo Alto, California: Tioga Press.

Pearl, Judea (1982) " On the discovery & generation of certain heuristics." The UCLA Computer Science Quarterly 10(2):121:132. Also appeared in the Artificial Intelligence Magazine Winter/Spring 1983 pp23-33.

Pearl, Judea (1984) "Heuristics, Intelligent Search Strategies for Computer Problem Solving" Addison-Wesely 1984.

Waltz, D., (1975) "Understanding line drawings of scenes with shadows," in the book The Psychology of computer vision, Winston P. H. (Ed.), McGraw-Hill, New York, 1975.