On the Parallel Complexity of Some Constraint Satisfaction Problems

Simon Kasif

Department of Electrical Engineering and Computer Science
The Johns Hopkins University

ABSTRACT

Constraint satisfaction networks have been shown to be a very useful tool for knowledge representation in Artificial Intelligence applications. These networks often utilize local constraint propagation techniques to achieve global consistency (consistent labelling in vision). Such methods have been used extensively in the context of image understanding and interpretation, as well as planning, natural language analysis and commonsense reasoning. In this paper we study the parallel complexity of discrete relaxation, one of the most commonly used constraint satisfaction techniques. Since the constraint propagation procedures such as discrete relaxation appear to operate locally, it has been previously believed that the relaxation approach for achieving global consistency has a natural parallel solution. Our analysis suggests that a parallel solution is unlikely to improve by much the known sequential solutions. Specifically, we prove that the problem solved by discrete relaxation is log-space complete for P (the class of polynomial time deterministic sequential algorithms). Intuitively, this implies that discrete relaxation is inherently sequential and it is unlikely that we can solve the polynomial time version of the consistent labelling problem in logarithmic time by using only a polynomial number of processors. Some practical implications of our result are discussed.

1. Introduction

Constraint satisfaction networks have been shown to be a very useful tool for knowledge representation in Artificial Intelligence applications [Winston 84]. These networks often utilize local constraint propagation techniques to achieve global consistency. Such methods have been used extensively in the context of image understanding and interpretation [Rosenfeld et al. 76], [Haralick & Shapiro 79], [Mackworth 77] as well as planning, natural language analysis and commonsense reasoning [Winston 84].

In particular, this paradigm has been applied to solve the *consistent labelling problem* (CLP) which is a key problem in many computer vision applications. The consistent labeling problem can be informally defined as follows. Let S be a set of objects. Each object has a set of possible labels associated with it. Additionally, we are given a set of constraints that for each object s and label x describe the compatibility of assigning the label x to object s with assignment of any other label x' to any other object s' .

Since CLP is known to be NP-complete, the discrete relaxation method has been proposed to reduce the initial ambiguity. The Relaxed Consistent Labeling Problem (RCLP) allows an assignment of a label x to an object s iff for any other object s' in the domain there exists a valid assignment of a label x' which does not violate the constraints (a formal definition is given in the next section). This formalization allows us to achieve global consistency by local propagation of constraints. Specifically, a discrete relaxation algorithm can discard a label from an object if it is incompatible with all other possible assignments of labels to the remaining objects. The discrete relaxation approach has been successfully applied to numerous computer vision applications [Waltz 75.], [Kitchen 1980], [Barrow & Tenenbaum 76], [Brooks 81]. The sequential time complexity of RCLP is discussed in [Mackworth & Freuder 85].

In this paper we study the parallel complexity of RCLP. Since the constraint propagation procedures such as discrete relaxation appear to operate locally, it has been previously believed that the relaxation approach for CLP has a natural parallel solution [Rosenfeld et al. 76], [Ballard & Brown 82], [Winston 84]. Our analysis suggests that a parallel solution is unlikely to improve by much the known sequential solutions. Specifically, we prove that the relaxed consistent labelling problem belongs to the class of inherently sequential problems called log-space complete for P.

Intuitively, a problem is log-space complete for P iff a logarithmic time parallel solution for the problem will produce a logarithmic time parallel solution for any polynomial time deterministic sequential algorithm. This implies that unless P $\subseteq$ NC (the class of problems solvable in logarithmic parallel time with polynomial number of processors) we cannot solve the problem in logarithmic time using a polynomial number of processors. This result is based on the "parallel computation thesis" proved in [Goldschlager 78] that establishes that parallel time computation is polynomially related to sequential space. Specifically, the class of problems that

can be solved in logarithmic parallel time with polynomial number of processors is equivalent to the class of problems that can be solved in polynomial time using logarithmic space on a sequential machine. For length considerations, we assume that the reader is familiar with elementary complexity theory and log-space reducibility techniques [Garey & Johnson 79] and the literature on discrete relaxation (network consistency algorithms). For completeness we shall provide the necessary definitions in the next two sections.

2. Consistent Labelling Problems and Discrete Relaxation

The consistent labelling problem (CLP) and its less restrictive (relaxed) version are formally defined in [Mackworth 77] and [Rosenfeld et al. 76]. For completeness we give a semiformal definition here. Let $V=\{v_1, \ldots, v_n\}$ be a set of variables. With each variable v_i we associate a set of labels L_i. Now let P_{ij} be a binary predicate that defines the compatibility of assigning labels to objects. Specifically,

$$P_{ij}(x,y)=1$$

iff the assignment of label x to v_i is compatible with the assignment of label y to v_j. The Consistent Labelling Problem (CLP) is defined as the problem of finding an assignment of labels to the variables that does not violate the constraints given by P_{ij}. More formally, a solution to CLP is a vector $(x_1, \ldots, x_n)$ such that x_i is in L_i and for each i and j, $P_{ij}(x_i, x_j) = 1$.

For example, the 4-queens problem can be seen as an instance of CLP. To confirm this, associate a variable with each column in the board and let $L_i=\{1,2,3,4\}$ for $1 \leq i \leq 4$. Let $P_{ij}(x,y)=1$ iff positioning of a queen in row x at column i is " safe " when there is a queen in column j and row y.

As mentioned in the introduction, the CLP is known to be NP-complete. Therefore several polynomial approximation algorithms were proposed and were shown to perform quite well in practical applications. The most significant class of algorithms are variations on discrete relaxation [Rosenfeld et al. 76] also known as network consistency algorithms [Mackworth 77]. Formally, a solution to the relaxed version of CLP (RCLP) is a set of sets $M_1, \ldots, M_n$ such that M_i is a subset of L_i and a label x is in M_i iff for EVERY M_j $i \neq j$ there is a y_{xj} in M_j, such that $P_{ij}(x, y_{xj}) = 1$. Intuitively, a label x is assigned to a variable iff for every other variable there is at least one valid assignment of a label to that other variable that supports the assignment of label x to the first variable. Clearly, any solution to CLP is also a solution to RCLP but not vice versa. In this sense discrete relaxation is a form of incomplete limited reasoning. We call a set $M_1, \ldots, M_n$ to be a maximal solution for a RCLP iff there does not exist any other solution $S_1, \ldots, S_n$ such that $M_i \subseteq S_i$ for all $1 \leq i \leq n$. We are only interested in maximal solutions for a RCLP. This restriction is necessary since any RCLP has a trivial solution: the set of empty sets. Additionally, recall that any solution for a RCLP represents a set of candidate solutions for the original CLP, which will eventually be verified by a final exhaustive check. Thus, by insisting on maximality we guarantee that we are not loosing any possible solutions for the original CLP. Therefore, in the remainder of this paper a solution for a RCLP is identified with a maximal solution.

3. The Complexity of Searching AND/OR Graphs

In this section we state several preliminary definitions and results that will be used in the following section to analyze the complexity of RCLP. We begin by defining AND/OR graphs [Nilsson 71].

An AND/OR graph is a 6-tuple (A, O, E, s, S, F) where A is a set of AND-nodes, O is a set of OR-nodes, E is a set of directed edges connecting nodes in $A \cup O \cup S \cup F$, s is a unique start node in A, S is a set of success nodes and F is a set of failure nodes. The solvability of a node in an AND/OR graph is defined recursively:

- If x is a S-node, it is solved.
- If x is is an AND-node, then it is solved iff all its successors (defined by the direction of the edges of E) are solved.
- If x is an OR-node then it is solved iff one of its successors is solved.

An AND/OR graph has a solution iff s is solved.

Proposition 3.1: (Jones & Laaser)
Finding a solution for an AND/OR graph is log-space complete.

Proof:

This result can be obtained by observing that GAME studied in [Jones & Laaser 77] is an instance of the problem of AND/OR solvability.

We now define the class of propositional Horn clauses.

A propositional formula H is said to be a propositional Horn clause iff one of the following holds

- H is a propositional atom of the form Q, called an assertion.
- H is a propositional formula of the form $P \leftarrow P_1 \& , \ldots, \& P_n$, denoted by $P \leftarrow P_1, \ldots, P_n$ and called an implication.
- H is a propositional negative atom (literal) of the form $\neg P$, denoted by $\leftarrow P$ and called a goal.

We note that our slightly restrictive definition of Horn clauses (not allowing multiple literals in the goal) does not restrict the expressiveness of the language.

A *Propositional Logic Program* is a set of propositional Horn clauses with a single goal. We define the unsatisfiability of a set of propositional Horn clauses S as a solvability relation on the set of propositional names in S. The definition is recursive:

- If P is an assertion in S then it is solvable.
- If P appears on the left hand side in a set of implications of the form

$$P \leftarrow P_1, \ldots, P_n$$

then P is solvable iff each one of the P_is is solvable in at least one of the implications.

A propositional logic program is *unsatisfiable* iff the propositional name that appears in the single negative atom is solvable. The problem of testing whether a propositional logic program is unsatisfiable will be referred to as the Propositional Horn Satisfiability Problem (PHSP).

Example:

The following set of Horn clauses is unsatisfiable since P is solvable:

$\leftarrow$ P
P $\leftarrow$ Q,R.
P $\leftarrow$ S,T.
R $\leftarrow$ S.
T $\leftarrow$ P.
Q.
S.

The next theorem, though not explicitly stated previously in a published form, is part of the common folklore among theoreticians [Ullman 85].

Theorem 3.1 (folklore):

The problem of testing the satisfiability of propositional Horn clauses is log-space complete.

Proof:

The proof is by reduction from solvability of AND/OR graphs (GAME of [Jones & Laaser 77]) and will not be presented here in full detail. Generally, "reduction" is the most common technique to show a problem X is log-space complete. Specifically, it is adequate to show the problem is in P (the class of polynomial time algorithms), and then reduce a known log-space complete problem to X using a function computable in logarithmic space (log-space) by a deterministic Turing machine. Since log-space reducibility is a transitive relation, we can then deduce that if we had a logarithmic time parallel algorithm to solve X, we could also perform every other sequential polynomial time computation in logarithmic time. In our case the reduction of AND/OR graph solvability to propositional Horn satisfiability is immediate (in some sense it is the same problem). We label all the nodes in the AND/OR graph with distinct propositional atoms. Then for each AND-node P connected to $P_1, \ldots, P_n$ we create a formula $P \leftarrow P_1, \ldots, P_n$. For each OR-node P connected to $P_1, \ldots, P_n$ we create formulae

$$P \leftarrow P_1.$$

$$\ldots$$

$$P \leftarrow P_n.$$

For each terminal success node Q we create the assertion Q. Finally if the start node of the graph is labelled by P we add the goal $\leftarrow$ P to the set.

It is easy to see that the original graph has a solution iff the set of formulae created in this fashion is unsatisfiable and the transformation can indeed be done in log-space.

It is not difficult to verify that the following is also true [Reif 85]:

Theorem 3.2:

Theorem 3.1 holds for propositional logic programs restricted to implications that have at most two atoms on the right hand side of the implication.

4. The Complexity of RCLP

In this section we prove our main result, namely that the problem of finding a solution to Relaxed CLP (RCLP) is log-space complete. To accomplish this we show that RCLP is in P and subsequently prove that satisfiability of propositional Horn clauses (PHSP) is reducible to RCLP. The first part of the proof is straightforward since most existing algorithms for RCLP are of polynomial sequential complexity (see [Mackworth & Freuder 85]). In fact the edge consistency algorithm as given in [Mackworth 77] is linear in the number of edges in the constraint graph [Mackworth & Freuder 85].

Theorem 4.1:

The Propositional Horn clause satisfiability problem is log-space reducible to the Relaxed Consistent Labelling Problem.

Proof:

Let Pr be a propositional logic program such that no implication has more then two atoms on its right hand side. We will also assume that all the atoms in Pr are uniquely labeled with integer values. We shall construct an RCLP G from Pr such that Pr is unsatisfiable iff a unique variable $<P_0>$ in G that corresponds to the unique goal $\leftarrow P_0$ does not have a valid assignment of the label f. The RCLP is constructed in the following way.

1. For each atom in Pr A we create a unique variable <A>.
2. For each assertion Q in Pr we create a unique variable $<SOLVED_Q>$.

3. Create a unique variable $<P_0>$ that corresponds to the goal $\leftarrow P_0$ of Pr.
4. For each implication of the form $P \leftarrow Q, R.$ we add the variable <Q,R> to G.

This construction defines all the variables of G. The initial label sets are created as follows:

- Each variable with the exception of the <SOLVED> variables is assigned the label l.
- For each assertion Q in Pr we add the label q to the initial label set of $<SOLVED_Q>$.
- For each variable of the form <R> we add the label f to its initial set.
- For each variable of the form <S,T> we add the labels f_S and f_T to its initial set.

We are now ready to define the constraints of the problem G. We define the constraints using a compatibility matrix COM, whose entries are of the form COM[variable,variable,label, label]. COM[v_i, v_j, x, y] = 1 iff the assignment of label y to variable v_j is compatible with the assignment of label x to variable v_i. An alternative natural representation is to use a directed multigraph where the nodes correspond to the variables of the problem and the edges are labeled with the constraints of the problem. It is important to observe that in order to preserve log-space reducibility we do not need to create the entire compatibility matrix COM. For a full description of the RCLP we only need to create a list of the constraints of the form COM[var,var,label,label] = 0. That is, we describe only the incompatible assignments. The remaining entries in the matrix can be filled with 1s.

For each implication of the form $P \leftarrow R$ we add the constraints

COM[<P>,<R>,f ,f] = 1
COM[<P>,<R>,f ,l] = 0

For each implication of the form $P \leftarrow Q, R$ we add the constraints

COM[<P>,<Q,R>,f ,f_Q] = 1
COM[<P>,<Q,R>,f ,f_R] = 1
COM[<P>,<Q,R>,f ,l] = 0

COM[<Q,R>,<R>,f_R ,f] = 1
COM[<Q,R>,<R>,f_R ,l] = 0

COM[<Q,R>,<Q>,f_Q ,f] = 1
COM[<Q,R>,<Q>,f_Q ,l] = 0

Finally, for every variable of the form $<SOLVED_Q>$ we add the constraint

COM[<Q>, $<SOLVED_Q>$,f ,q] = 0

This completes the definition of all the "necessary" constraints of the the RCLP. The rest of the matrix COM can be filled with 1s.

Note that the label f must be removed from all the variables that correspond to the assertions of the logic program. Using induction on the length of the satisfiability proof it is fairly easy to show that the label f will be removed from the variable $<P_0>$ that corresponds to the goal $\leftarrow P_0$ iff P_0 is solvable. The formal proof is omitted.

Now we have to verify that the above construction can be done using only logarithmic space on the work tape of the Turing machine. We shall sketch the main ideas of the proof method. If the construction were to be carried out in the order given above it would have taken linear space (linear in the number of total occurrences of all the atoms in Pr). Fortunately, since we assumed the atoms were initially numbered by integers, we can follow the above construction in a demand-driven fashion as explained below.

To start off we can create all the variables of the form <Q> and their respective label sets. This can be done with logarithmic space consumption since processing each one of the N-variables we need lgN-bits. For each assertion we can add the respective label to <SOLVED>. For implications of the form $P \leftarrow Q, R$ we generate a new variable and its respective initial label set. This step requires a counter that can be implemented in logarithmic space. Finally, for each implication encountered we can generate the constraints (again in logarithmic space). This completes the generation of all the necessary (see above discussion) information that completely describes the RCLP G.

Example:

Consider the following PHSP :

$\leftarrow$ P
P $\leftarrow$ Q,R.
P $\leftarrow$ S,T.
R $\leftarrow$ S.
T $\leftarrow$ P.
Q.
S.

We construct the following RCLP. The variables of the problem are: <P>, <Q>, <R>, <S>, <T>, <Q,R>, <S,T>, $<SOLVED_Q>$, $<SOLVED_S>$. The initial assignments of labels are as follows:

<P>: {f ,l }
<Q>: {f ,l }
<R>: {f ,l }
<S>: {f ,l }
<T>: {f ,l }
<Q,R>: {f_Q ,f_R ,l }
<S,T>: {f_S ,f_T ,l }
$<SOLVED_Q>$: {q }
$<SOLVED_S>$: {s }

Finally, the constraints are given in Figure 1.

$$COM[<P>,<Q,R>,f\ ,l] = 0$$
$$COM[<P>,<S,T>,f\ ,l] = 0$$

$$COM[<Q,R>,<R>,f_R,l] = 0$$
$$COM[<Q,R>,<Q>,f_Q,l] = 0$$

$$COM[<S,T>,<S>,f_S,l] = 0$$
$$COM[<S,T>,<T>,f_T,l] = 0$$

$$COM[<R>,<S>,f\ ,l] = 0$$

$$COM[<T>,<P>,f\ ,l] = 0$$

$$COM[<Q>,<SOLVED_Q>,f\ ,q] = 0$$

$$COM[<S>,<SOLVED_S>,f\ ,s] = 0$$

Figure 1.

5. Conclusion

In this paper we have shown that a very important class of algorithms which were previously believed to be highly parallelizable are in fact inherently sequential. This negative result needs to be quantified. Essentially, it suggests that the application of massive parallelism will not change significantly the worst case complexity of discrete relaxation (unless one has an exponential number of processors). However, this result does not preclude research in the direction of applying parallelism in a more controlled fashion. Specifically, speedups are possible in the case where the number of processors is significantly smaller than the size of the constraint graph (a very likely case). In this case, it may be possible to obtain a full P-processor speedup. We are currently actively investigating this interesting case.

Acknowledgements

Thanks are due to Azriel Rosenfeld, Dave Mount and Deepak Sherlekar for their constructive comments that contributed greatly to the final form of this paper. This work was supported by NSF under grant DCR-18408 while the author was a visiting scientist at the Center for Automation Research, University of Maryland.

REFERENCES

[1] Ballard, D.H. and C.M. Brown, *Computer Vision,* Prentice Hall, 1982.

[2] Barrow, H. G. and J. M. Tenenbaum, MSYS: A system for reasoning about scenes, Technical Note 121, SRI AI Center, Menlo Park, CA, April 1976.

[3] Brooks, R A., Symbolic reasoning among 3-D models and 2-D images, *Artificial Intelligence* **17**, pp. 285-348, 1981.

[4] Garey, M.R and D. S Johnson, *Computers and Intractability: A Guide to NP-Completeness,* Freeman, San Francisco, 1979.

[5] Goldschlager, L.M., A Unified Approach to Models of Synchronous Parallel Machines, *Proc. of the 10-th Symposium on Theory of Computing,* pp. 89-94, May 1978.

[6] Haralick, R. M. and L. G. Shapiro, The consistent labeling problem: Part I, *IEEE Trans. Patt. Anal. Mach. Intel.* **PAMI-1**, pp. 173-184, 1979.

[7] Jones, N. and T. Laaser, Complete problems for deterministic polynomial time, *Theoretical Computer Science* **3**, pp. 105-117, 1977.

[8] Kitchen, L. J., Relaxation applied to matching quantitative relational structures, *IEEE Trans. Syst. Man Cybern.* **SMC-10**, pp. 96-101, 1980.

[9] Mackworth, A. and E. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction, *Artificial Intelligence* **25**, pp. 65-74, 1985.

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

[11] Nilsson, N.J., *Problem-Solving Methods in Artificial Intelligence,* McGraw-Hill, New York, 1971.

[12] Reif, J., Depth-first search is inherently sequential, *Info. Proc. Letters* **20**, pp. 229-234, 1985.

[13] Rosenfeld, A., R. Hummel, and S. Zucker,, Scene labeling by relaxation operations, *IEEE Trans. Syst. Man Cybern* **SMC-6**, pp. 420-433, 1976.

[14] Ullman, J., Personal communication. 1985.

[15] Waltz, D., Understanding line drawings of scenes with shadows, pp. 19-92 in *The Psychology of Computer Vision,* ed. P. H. Winston, McGraw-Hill, New York, 1975..

[16] Winston, P.H., *Artificial Intelligence,* Addison Wesley, 1984.