TR79-13

Distributed Problem Solving Using Iterative Refinement

R.S.Brooks and V.R.Lesser

Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

May 1979

Abstract

The research presented in this paper concerns the development of a distributed iterative refinement algorithm for network traffic light control. Many distributed problem solving applications, including distributed network traffic light control, exhibit a natural spatial distribution of sensors and/or effectors. In order for these types of applications to be distributed effectively, processing should reflect the natural spatial distribution. Often this will require processors to operate on local data-bases that are incomplete and possibly inconsistent because the cost of maintaining complete and consistent data-bases can be prohibitive.

The iterative refinement technique appears well suited for this type of distributed problem solving because it supports the desired problem decomposition and can operate on incomplete, local data bases. Two general classes of iterative refinement algorithms are examined: single-label and multi-label. Members of the first class of algorithms resemble classical "hill-climbing" algorithms; members of the second class of algorithms resemble "relaxation" algorithms which are often used in image processing applications. The various algorithms developed are described, results of experiments with the algorithms are presented, and error and uncertainty in the algorithms is discussed.

This research was supported in part by NSF grant MCS78-04212 to the University of Massachusetts.

Table of Contents

1. Introduction

Distributed problem solving is a new area for research in artificial intelligence (AI) which involves performing such AI problem solving tasks as signal-interpretation [Erman and Lesser 1975; Riseman and Arbib 1977; Hanson and Riseman 1978b], planning [Sacerdoti 1977; Tate 1977] etc. in a distributed processing environment. In a distributed problem solving system, the the goal of the decomposition is to permit parallel operation of processors on different parts of the problem with limited sharing of information, limited synchronization, and distributed control.

Many applications such as network traffic light control, distributed sensor networks, distributed air traffic control, etc., exhibit a natural spatial distribution of sensors and effectors. In order for these types of applications to be distributed effectively, processing should reflect this natural spatial distribution. Not all problem solving techniques can be replicated directly or partitioned based on the distribution of the sensors and effectors. Rather, it may be necessary to modify the techniques so that they operate on local data-bases that are incomplete and possibly inconsistent, due to the communication and synchronization costs of maintaining complete and consistent information. For some techniques (e.g., A*, dynamic programming, etc.), such modifications are difficult or impossible because of their reliance on complete and consistent information. However, other techniques (e.g., relaxation - a form of iterative refinement, hypothesize-test, etc.) appear to be more easilly transferrable to a distributed problem solving environment because of their ability to function with incomplete and inconsistent information.

The research presented here analyzes the suitability of a general problem solving technique, Iterative Refinement (IR), for distributed problem solving. Using the IR approach, a problem is decomposed into subproblems, each of which is solved using limited information. The set of subproblem solutions (partial solutions) defines the overall solution.

The limited information used in solving subproblems comes from using only local (sensory) data and the "tentative" solutions to a small subset of other subproblems. These other subproblems are chosen because the solution to the desired subproblem is strongly dependent on their solution. Usually, these strongly dependent subproblems are neighboring subproblems; that is, they are physically adjacent with respect to the spatial distribution of sensors and effectors. This is advantageous for a distributed system because it limits communication to neighboring processors.

Subproblem solutions are tentative since they may be incorrect due to limited information. Incorrect subproblem solutions are corrected through repetitive processing. In each repetition, missing information necessary for the correct answer to a subproblem is acquired indirectly through the refinement of interdependent subproblem solutions. This iterative refinement process continues until a fixed-point (i.e., a solution for which there are no further refinements) has been found.

There are a number of classes of IR algorithms, reflecting different forms of partial solutions, update (refinement) functions, and control schemes. A partial solutions, of a single-label IR algorithm is a single value (a label); the update function selects a new label on the basis of local context and neighbors' current labels. A partial solution of a multi-label IR algorithm is a probability distribution on the possible labels of the partial solution. This type of IR is called relaxation, and its update function produces new probabilty distributions for partial solutions on the basis of local context and neighbors' probability distributions. For both types of IR algorithm, a control scheme determines when and where to make refinements. This may involve serial or parallel refinements to different parts of the solution utilizing local and/or global rules.

Work on using IR for distributed problem solving is just beginning. Baudet's work on asynchronous iterative methods [Baudet 1976 and 1978] is encouraging, but is limited to problems involving continuous operators with unique fixed-points. Unfortunately, many real world applications are formulated in terms of discontinuous operators with multiple fixed-points. The literature on image processing includes many applications of relaxation [Waltz 1975; Zucker, Hummel, and Rosenfeld 1975; Hanson and Riseman 1978a] which could operate on parts of an image in parallel if implemented on an appropriate parallel processing system. However, in these applications dependencies between non-neighboring subproblems are weak. Again, some real world applications cannot be structured so as to avoid strong dependencies among non-neighboring subproblems.

In order to test the suitability of IR using a real world application, we have chosen to study the problem domain of Network Traffic Light Control (NTLC); NTLC problems are familiar and easily understood, have a natural spatial distribution, and exhibit strong dependencies between non-neighboring subproblems. In our formulation of a distributed NTLC system, every intersection with signals is equipped with sensors for detecting incoming traffic volumes. Each intersection that is equipped with a signal also has a processor that controls the intersection's signals based on the local sensory data and on communication

with the processors of neighboring intersections.

The NTLC problem addressed in this paper is presented in more detail in Section 2. A centralized NTLC algorithm, SIGOP II [Lieberman and Woo 1976], which provides the basis for our research is described in Section 3. Section 4 describes our research on single-label IR. The approach involves the adaptation of SIGOP II for distributed NTLC. In adapting the algorithm for a distributed environment, certain approximations must be made to limit communication and computational requirements. A number of algorithms have been developed to explore the tolerance of the IR algorithms to uncertainty introduced by the use of these approximations.

A multi-label IR, or relaxation, approach to distributed NTLC is discussed in Section 5. This approach uses the same basic knowledge as that used by the single-label IR algorithms. Conclusions and a discussion of future directions for research follow in Section 6.

2. The Problem Domain: Network Traffic Light Control

Network traffic light control involves controlling signals so as to optimize traffic flow in a network with signals at each intersection [DOT/FHA 1976; Wagner et al 1971; Lieberman et al 1974a,b]. The research presented in this paper concerns the problem of finding an optimal* network timing plan for a network under moderate traffic conditions. As traffic conditions change, new timing plans are determined and implemented.

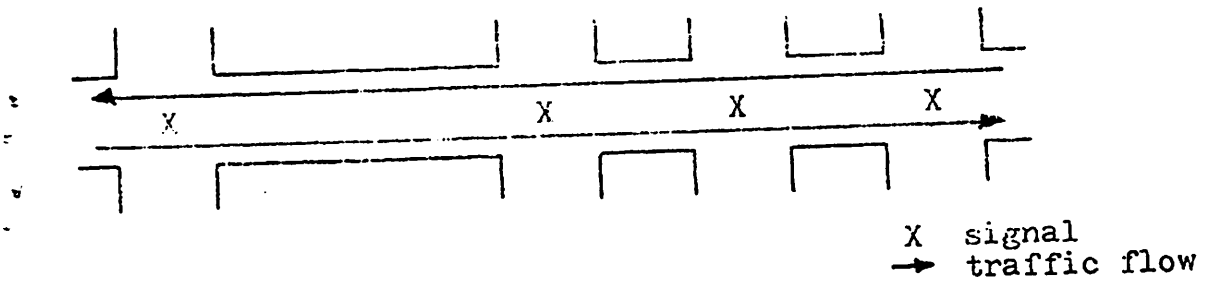
An optimal network timing plan for the moderate flow regime maximizes traffic flow on major arteries of the network by appropriate traffic signal settings which minimize stops and delay incurred by vehicles traversing the arteries. A key assumption used in developing an optimal network signal control pattern for moderate traffic flows is that vehicles travelling along roads (links) between intersections (nodes) are released together in platoons from a signal, and travel together in platoons to the next signal.

The use of a platoon-based traffic flow model can be represented graphically with a time-space diagram, as shown in Figure 1. Several important features are illustrated in the figure. Progressive movement of platoons in the network

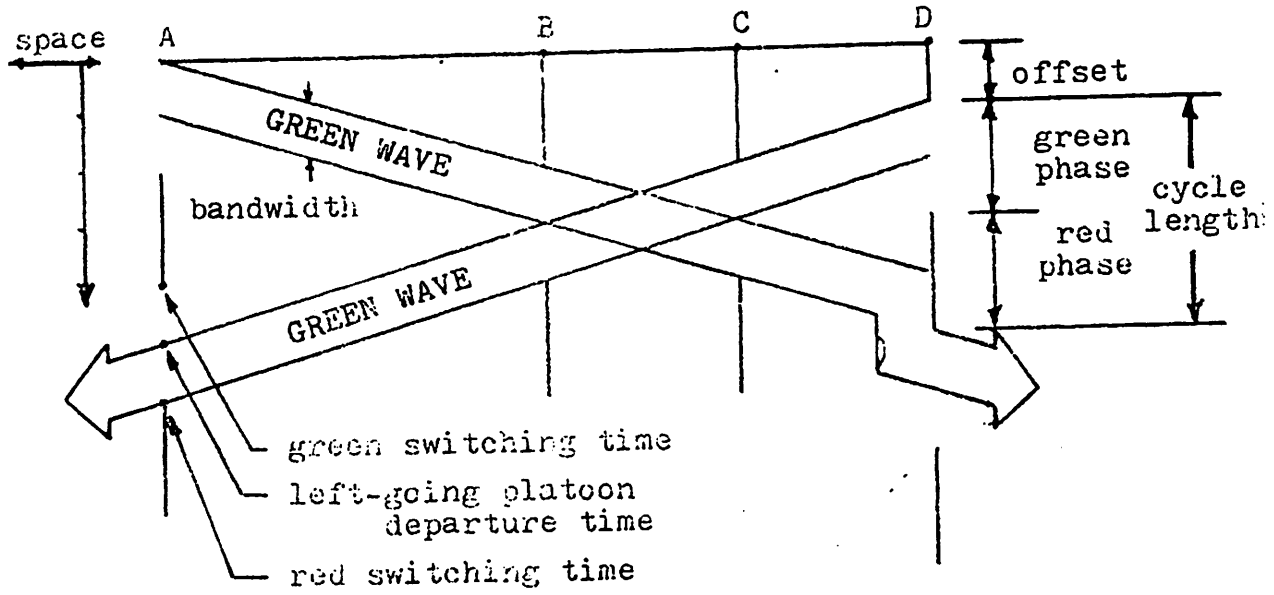
---

* Optimality is based on "steady-state" conditions (i.e., transient behavior resulting from changes in timing plans has died out and a steady state has been achieved).

the Network:



Time-Space Diagram:



slope of green wave is proportional to platoon free-flow speed

Time - Space Diagram
Figure 1

is represented as green waves. The location of a green wave, also termed a green band or through band, is determined by the control settings of signals in the network and a set of rules governing the structure of the green wave. The bandwidth indicates the period of time availible for traffic flow within the green wave. The cycle length of a signal is the length of time in seconds for a complete sequence of signal phases. The split specifies what portion of the cycle is taken up by the green phase. An offset expresses the time relationship between the start of a cycle and a reference point common to all nodes.

The timing plan for a signal can be specified as a cycle length, a split, and an offset. An alternate specification consists of a cycle length and switching times which describe when within a cycle length a signal switches to the next phase. A network timing plan is made up of timing plans for all signals in the network. Our study concerns algorithms for computing network timing plans given a common cycle length and fixed splits for all signals of the network; thus, what must be solved for are offsets (or green switching times) for all signals.

Optimality of a network timing plan is determined on the basis of global disutility. Disutility is a measure of stops and delay incurred by traffic traversing links of the network. An optimal network timing plan minimizes global disutility, the sum of disutility of all individual links of the network. To simplify our study we have limited it to single artery networks with single (primary) platoon flows for each direction along the artery. This eliminates such problems as modeling turning traffic and multiple platoons, and coping with loops in flows.

Disutility incurred by traffic traversing an individual link depends on the control settings for the signals at each end of the link and the arrival times of platoons entering the link.* Since these platoon arrival times depend on the control settings and platoon arrival times for other signals, disutility on a link is, in general, a global function of all signal control settings. As will be seen in future discussions, this is a very important characteristic of the problem.

By using state variables which represent platoon departure times, the link disutility calculations can be performed using only local information. This leads to the problem specification depicted in figure 2. The major ramification of utilizing platoon departure times in

---

* The arrival time of a platoon at node j on link (i,j) is equal to the departure time for the platoon at node i on the link plus the free-flow travel time for platoons traversing the link from node i to node j.

$$\underset{sg}{\text{MIN}} \sum_{(i,j)\in L} D(sg_j, sr_j, td_{ij}+d_{ij}, p_{ij})$$

constants:

S is the set of all intersections, i, equipped with a signal; $1 \le i \le N$.

L is the set of all links, (i,j), where platoons flow from intersection i to intersection j; $L \subseteq SxS$.

C is the common cycle length.

d_{ij} is the free-flow delay for platoons on link (i,j).

p_{ij} is the bandwidth for platoons on link (i,j).

g_i is the duration of the green phase of the signal at intersection i.

control variables: $sg = sg_1, sg_2, \ldots, sg_N$

sg_i is the green switching time for the signal at intersection i; $0 \le sg_i < C$.

state variables:

td_{ij} is the platoon departure time for platoons leaving intersection i on link (i,j); $sg_i \le td_{ij} < sr_i$.

$$td_{ij} = \begin{cases} sg_i & \text{if } \neg\exists h \,|\, (h,i)\in L \wedge h\neq j \\ PS(sg_i, sr_i, td_{hi}+d_{hi}, p_{hi}) & \text{if } \exists h \,|\, (h,i)\in L \wedge h\neq j \end{cases}$$

auxillary variables:

sr_i is the red switching time for the signal at intersection i; $sr_i = sg_i + g_i$.

ta_{ij} is the platoon arrival time for platoons arriving at intersection j on link (i,j); $ta_{ij} = td_{ij}+d_{ij} \equiv sg_j + ((ta_{ij}-sg_j) \bmod C)$.

functions:

$D(sg_j, sr_j, ta_{ij}, p_{ij})$ is the disutility function:

$$D(sg_j, sr_j, ta_{ij}, p_{ij}) = \begin{cases} 0 & \text{if } sg_j \le ta_{ij} < sr_j - p_{ij} \\ \alpha(ta_{ij}+p_{ij}-sr_j)(C-sr_j+\beta) & \text{if } sr_j - p_{ij} \le ta_{ij} < sr_j \\ \alpha p_{ij}(sg_j+C-ta_{ij}+\beta) & \text{if } sr_j \le ta_{ij} < sg_j+C \end{cases}$$

$PS(sg_i, sr_i, ta_{hi}, p_{hi})$ is the platoon structure function:

$$PS(sg_j, sr_j, ta_{ij}, p_{ij}) = \begin{cases} ta_{ij} & \text{if } sg_j \le ta_{ij} < sr_j \\ sg_j & \text{if } sr_j \le ta_{ij} < sg_j+C \end{cases}$$

Problem Statement
Figure 2

specifying the problem is that the platoon departure times for the whole network, which depend on upstream signals, may need to be updated whenever any signal's setting in the network is altered. The state variables can be viewed as representing an environment which reacts to changes in signal settings. In order to asses the worth of changes to signal timing plans (control variables) and provide a consistent base for further changes, the effects on traffic flow (the environment) must be simulated by updating the platoon departure times (state variables).

3. The SIGOP II Algorithm

The traffic light control algorithms developed in this paper are based on a centralized network timing plan generation program for moderate traffic: SIGOP II [Lieberman and Woo 1976]. SIGOP II appeared to be the most feasible of a number of NTLC approaches [Little et al 1974; Kinney et al 1977]. The SIGOP II program attempts to find a network timing plan which minimizes network disutility using a flow model which represents traffic flow explicitly as platoons of vehicles traversing links in the network. Knowledge of the network geometry (link lengths) and platoon bandwidths for platoons of each network link (predicted on the basis of link volumes) is used to determine a network timing plan.

The centralized SIGOP II algorithm is structured as an iterative refinement algorithm in order to reduce the complexity of problem solving. The problem of finding a network timing plan which minimizes network disutility is broken up into unit problems which involve finding a signal's timing plan which minimizes local disutility. Local disutility is disutility incurred by vehicles traversing links of a "mini-network" which consists of an intersection, it's neighbors, and the connecting links. The overall optimization process proceeds as follows:

1. Determine the sequence of nodes along a maximal spanning tree (MST) of the network. This ordering lists nodes attached to high volume links before nodes attached to low volume links.*

2. Prime the control settings at all intersections to optimize traffic operations along this ordering. Also platoon departure times are

---

* The MST of a network consisting of only a single artery includes all network links and has exactly one or two branches. A sequence along any MST begins with a root node and proceeds along the branches to the leaf nodes.

Preservation of Platoon Structure Validity
at Site of Update
by Sweeping along Traffic Flows

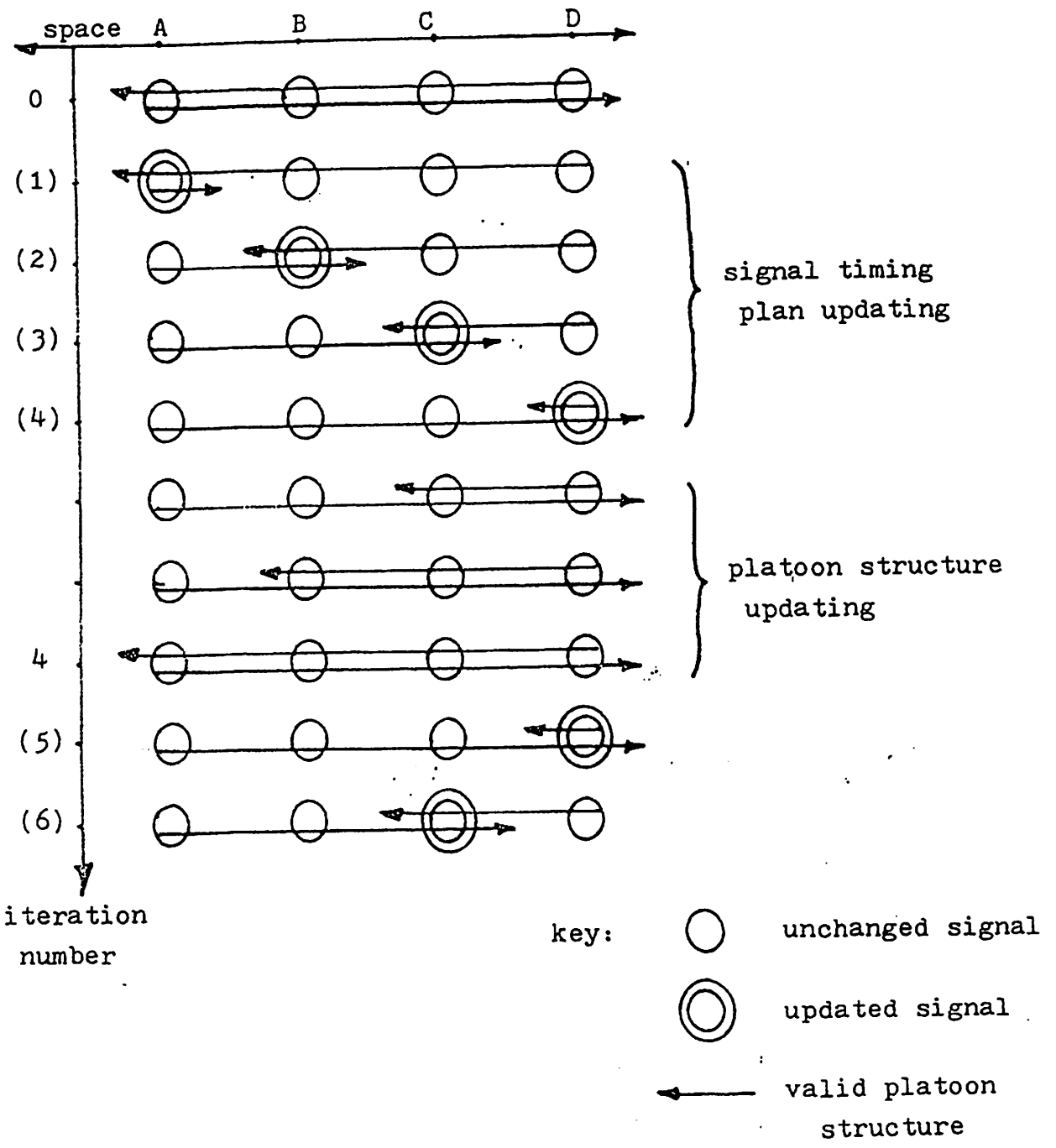


Figure 3

determined.

3. Reverse the sequence of nodes to get a new MST.

4. For each node, in the order indicated by the MST sequence, optimize the control setting based on local disutility and update local platoon departure times. In these calculations, the most recent values for signal settings and platoon departure times are used.

5. Calculate network-wide disutility (this entails updating platoon departure times until they are consistent, calculating local disutilities, and summing the local disutilities) and if zero or unchanged, stop.

6. If a predetermined number of sweeps over the network have been completed and convergence has not been achieved, select the best solution and stop, else continue with step 3.

The optimization is carried out by starting with an approximate solution (the prime) which has been obtained heuristically (steps 1 and 2). This priming process involves setting the signal at the first intersection arbitrarily and chosing settings for the remaining signals in the indicated sequence on the basis of local disutility, ignoring links attached to intersections where signal settings have not yet been determined. The prime is then repeatedly improved on by successive applications of local optimization (steps 3 through 6).

The use of a MST to determine a sequence for processing nodes is an important heuristic employed by the SIGOP II algorithm. The MST is determined based on traffic volumes; processing nodes along this spanning tree amounts to "sweeping" updates along the major flows of the network. This tends to preserve the validity of the platoon structure at the site of updates (see figure 3). Thus, updating of the network platoon structure can be eliminated after individual updates. Complete platoon structure updating, however, is still required periodically and is performed after each sweep to evaluate potential solutions because a valid platoon structure at all nodes is required to evaluate a network timing plan accurately.

Another, more subtle feature of the MST heuristic ordering is the reversing of the sweep direction after each sweep. This reversal, in addition to allowing each node to be updated exactly once on each sweep, aids in finding a good solution, as will be shown in section 3.2. It should be noted that each unit problem (local optimization) involves only variables of a node and its immediate

neighbors. This apparently makes the SIGOP II problem decomposition well suited for placing processors at intersections and utilizing neighbors-only communication.

3.1. Test Version Results

A SIGOP II test version was developed in order to provide a comparison for the distributed versions to be developed. The test version, SS, does not implement some features of SIGOP II which, for our purposes, would add unnecessary complexity. SS does not, for example, model secondary platoons and turning traffic. The tests, however, were conducted only on arterial networks where these features were relatively unimportant.

The performance of SS was measured on 6 six-intersection arterial configurations. Three test runs are depicted in Figures 4a,b, and c. The configurations include two variations of network geometry (intersection spacing) and three variations of platoon bandwidths (to simulate inbound, balanced, and outbound traffic flows). Optimal solutions were obtained for four test configurations, a close but suboptimal solution was obtained for one configuration, and a poor solution was obtained in one configuration. An average of two sweeps (one at minimum and four at maximum) was required to converge.

Additional tests on twelve-intersection arterial configurations were conducted to determine the algorithm's sensitivity to the network size. The quality of solution, measured as the average distance from optimal in units of disutility, was affected very little. The average number of sweeps to convergence was also changed little; however, sweeps were twice as long, effectively doubling the amount of computation required for convergence.

3.2. Additional Serial Versions for Comparison

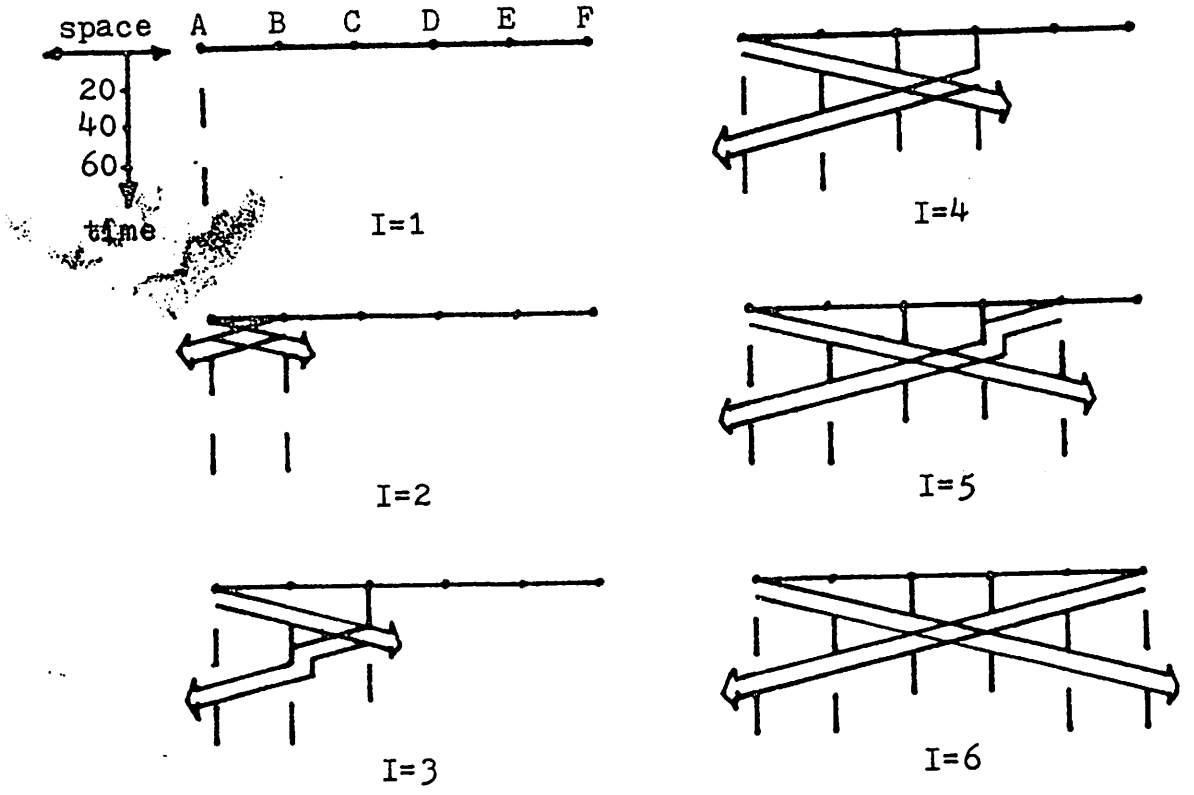
A number of modified SIGOP II test versions were developed to determine the importance of certain features of the basic SIGOP II algorithm. For example, SSP is a version that utilizes an arbitrary prime (as will be used by the distributed algorithms); results of tests on SSP indicate that the use of arbitrary primes for the simple arterial test cases does not significantly affect performance.

A number of features of SIGOP II's MST heuristics were tested with other versions called SRP, SRPE, SSPU, SSPA, and SSD. Versions SRP and SRPE compute updates in a random

FREE-FLOW DELAYS: AB=10, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT=10, RIGHT=10
****

ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	-	-	-	-	-	-	-
6	0	0	20	20	0	0	0
12	0	0	20	20	0	0	0

.
.
.



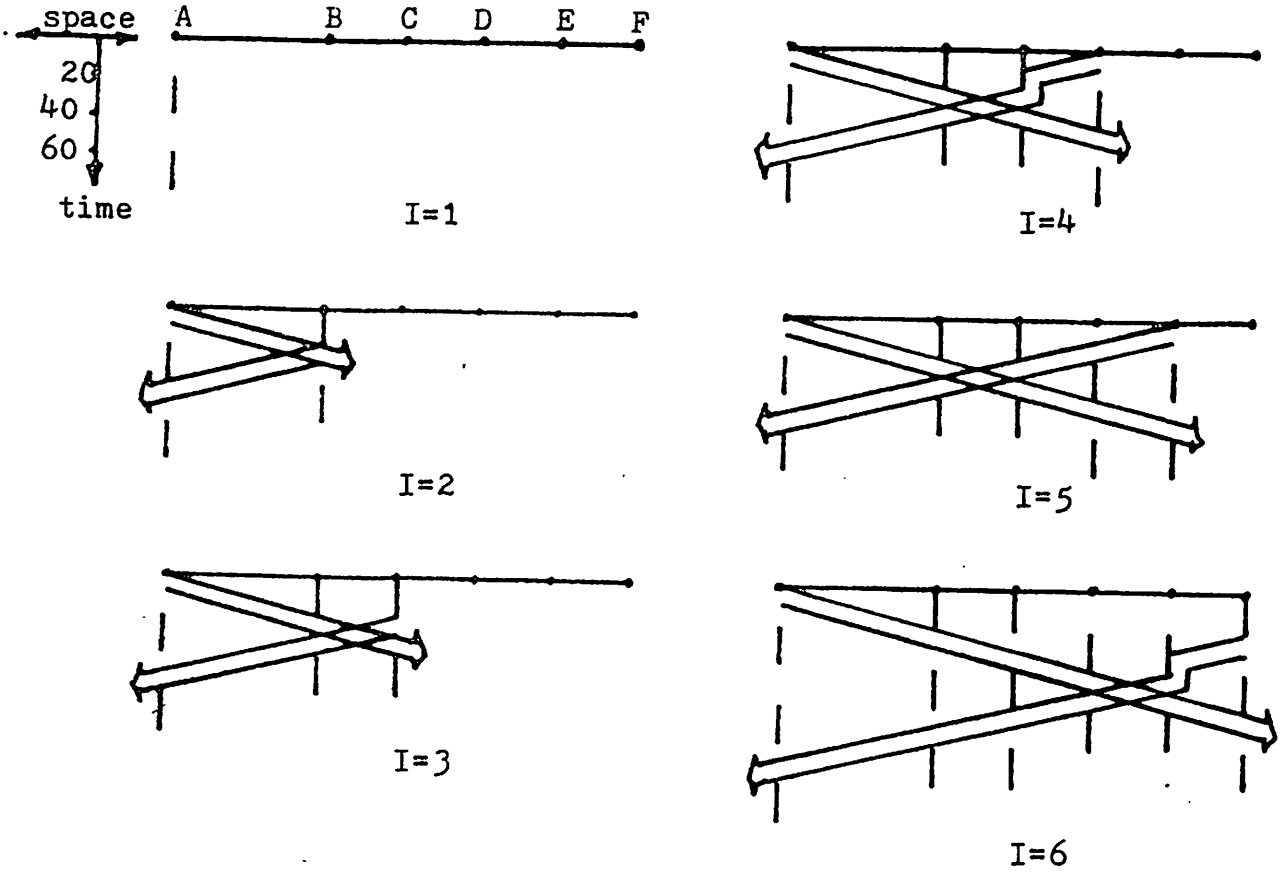
Run of SS, the SIGOP II test algorithm

Figure 4a

FREE-FLOW DELAYS: AB=20, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT=10, RIGHT=10
****

ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	-	-	-	-	-	-	-
6	0	20	20	0	0	20	28
12	0	20	20	0	0	20	28

.
.
.

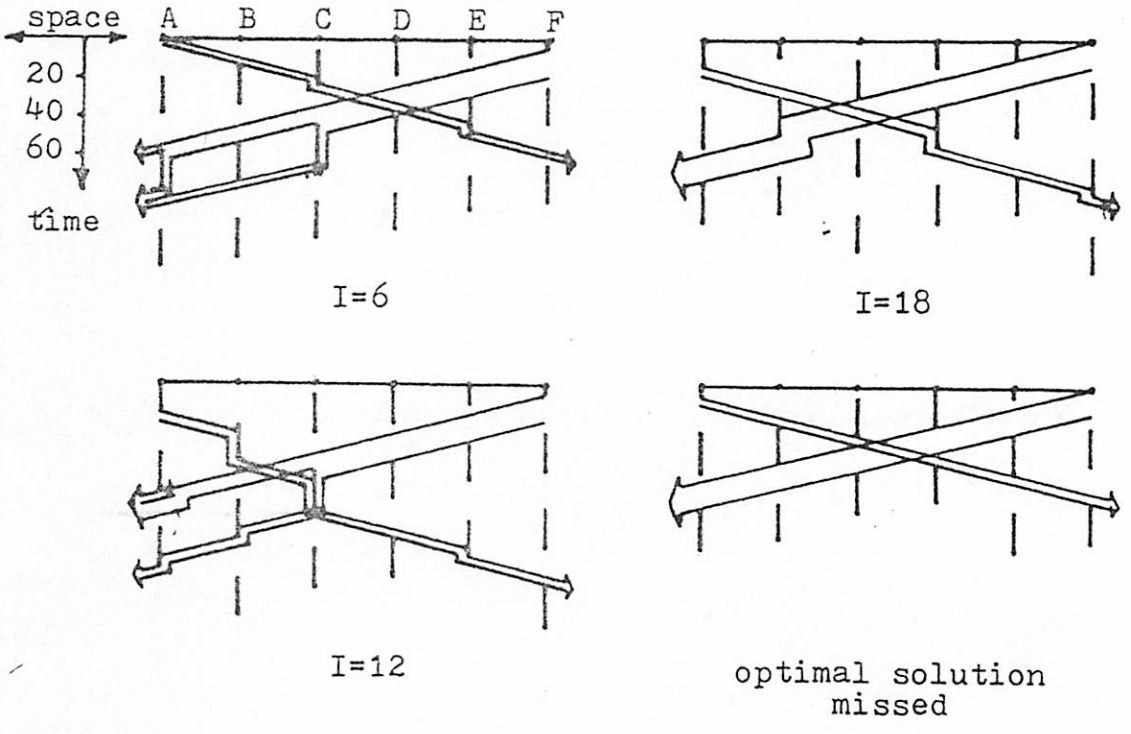


Run of SS, the SIGOP II test algorithm
Figure 4b

FREE-FLOW DELAYS: AB=10, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT=15, RIGHT= 5
****

ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	-	-	-	-	-	-	-
6	0	35	25	20	10	5	64
12	15	0	25	20	10	5	108
18	15	10	30	15	10	0	56
24	15	10	30	15	10	0	56

.
.
.



Run of SS, the SIGOP II test algorithm
Figure 4c

serial order rather than along the MST. SRPE differs from SRP in that extra platoon structure updating is performed between iterations to ensure accurate data for further updates. Version SSPU does not reverse the sweep direction after each sweep, and version SSPA skips updates that would increase global disutility. Results of tests on these versions, depicted in Figure 5, demonstrate that the many features of SIGOP II's MST heuristic node processing order are important.

Test version SSD determines the global worth of each node's proposed refinment, selects and implements the best single refinement, and then completely updates the platoon structure before determining the next refinement. This "steepest serial descent" algorithm is very stable because refinements can never decrease global goodness. This happens also to be true for version SSPA which was mentioned in the previous paragraph. Results of these versions seem to indicate that a small number of refinements which result in a decrease in global goodness can be tolerated and can even be beneficial in single-label IR algorithms.

4. Distributing SIGOP II: Single-label Iterative Refinement

In order to understand the effectiveness of the IR approach, it is important to review in what way partial solutions are approximate and how iterative refinement of approximate partial solutions leads to "correct" partial solutions. A partial solution is correct if it is part of an overall solution that maximizes some global metric of goodness (or minimizes a metric of disutility). Partial solutions are approximate when limited and possibly incorrect information is used to construct them. The use of this type of information leads to uncertainty as to which of the set of possible solutions for a subproblem will contribute to an overall solution that maximizes the global metric (i.e., an optimal solution).

One way of categorizing the types of uncertainty in partial solutions comes from analyzing the set of assumptions used for a particular algorithm. When assumptions may be violated, there is uncertainty as to the validity of computations based on these assumptions. These uncertainties can affect two major aspects of an IR algorithm: its stability (resistance to oscillation) and its ability to find an optimal or nearly optimal solution when multiple solutions exist.

The basic computation in an IR algorithm involves the updating of single subproblem solutions. Two assumptions are commonly employed to limit the information needed for updates. First, an update which increases goodness locally

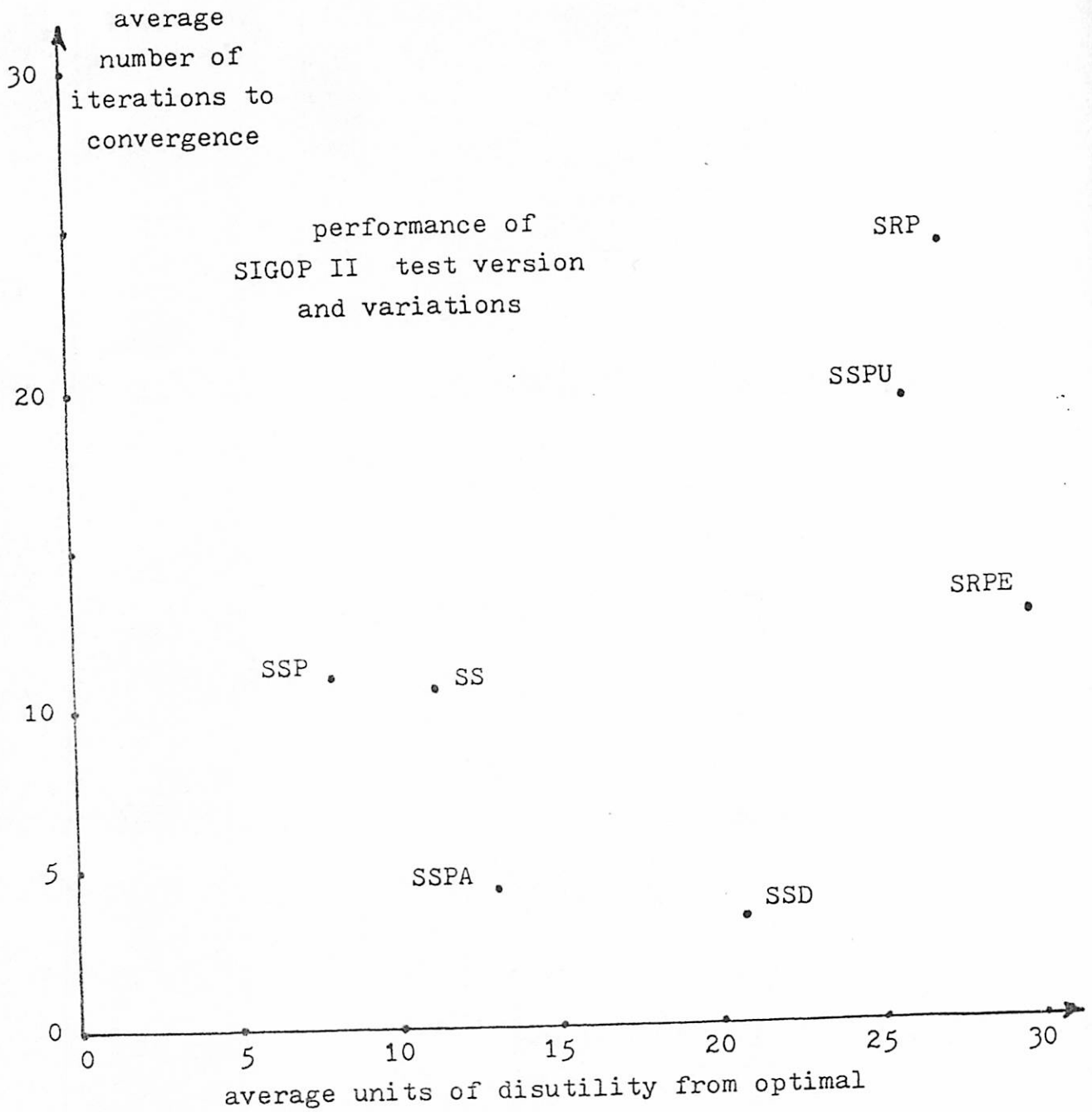
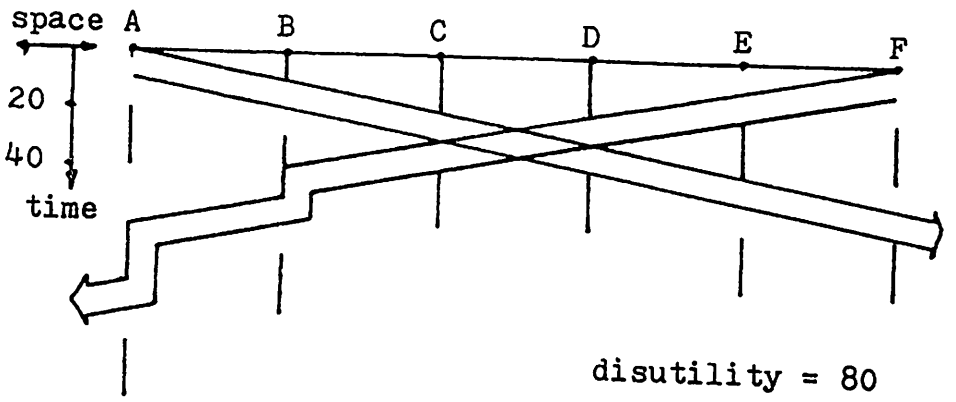
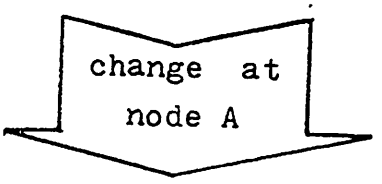
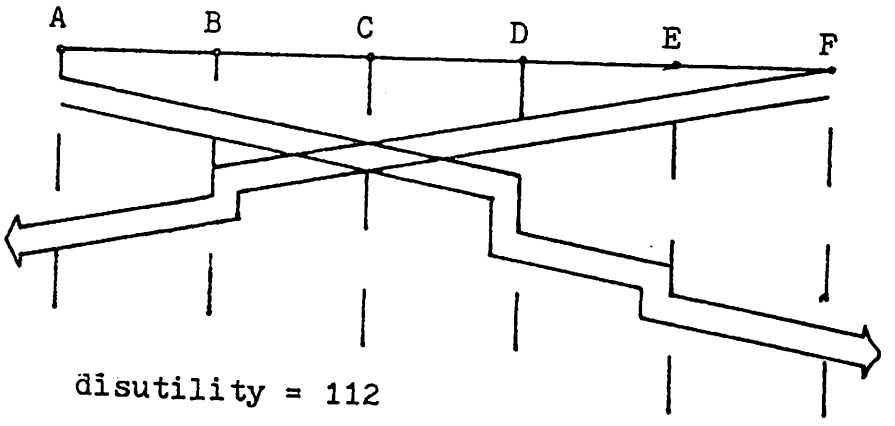

average
number of
iterations to
convergence
30
20
10
5
0
performance of
SIGOP II test version
and variations
SRP
SSPU
SRPE
SSP
SS
SSPA
SSD
0
5
10
15
20
25
30
average units of disutility from optimal

Figure 5







Local Improvement at A Results in an increase of global disutility

Figure 6

VERSION	DESCRIPTION	UNCERTAINTIES		
		LVU	SUU	EUU
SS	SIGOP II test version	E		*
SSP	SS but arbitrary prime	E		*
SRP	SSP but random order for updates	E		E
SRPE	SRP with extra environmental updating	E		
SSPU	SSP but unidirectional sweep	E		*
SSPA	SSP but skips updates which increase global disutility	C		*
SSD	SSP but picks globally best single update on each sweep	C		
PP	pure parallel test version	E	E	
PML	PP with modulation by fixed limit	RM	RM	
PMLA	PML improved	RM	RM	
PMP-x	PP with modulation by x percent	RM	RM	
PRC-x	PP but probability of a node being allowed to change of x	RF	RF	
PDMST	PP with MST heuristic	RF	RF	
PDMSTA	PDMST improved	RF	RF	
PPESS-x	PP with embedded SS to extend size of neighborhood to encompass x nodes on each side of a node	RMF	RMF	
PPGV	PP with global views		E	
PMLGV	PML with global views		RM	
SSPGV	SSP with global views			
PSD	PP with global best combination of changes determined and selected	C	C	
PSDML	PSD with modulation by fixed limit	C	C	
PMLSD	approximation to gradient descent	C	C	

uncertainties:

LVU - Local-View Uncertainty
SUU - Simultaneous-Update Uncertainty
EUU - Environmental-Update Uncertainty

key:

E - Exists
* - exists for networks with loops only
C - Compensated for in some way
RM - Reduced in Magnitude
RF - Reduced in Frequency
RMF - Reduced in both Magnitude and Frequency

Table 1

NAME		PERFORMANCE		PARALLELISM		SENSITIVITIES			
	!	D	I	P1	P2	VDP	VIP	VDS	VIS
SSP		7.9	11.1	0.5	0	5.2	1.0	3.5	11.5
PP	*	101.3	6.7	5.2	99.2				
PML		15.9	13.4	4.7	99.3	12.1	8.0	0.3	3.4
PMLA		12.6	15.3	4.3	88.9	9.6	9.4	0.3	2.9
PMP -25	*	30.9	25.8	5.3	100	29.3	15.8	11.6	8.7
PMP -40	*	23.4	24.5	4.7	98.0	8.5	12.2		
PMP -50	*	25.4	21.7	4.4	99.1	3.4	19.6	24.3	18.9
PMP -60		30.6	23.1	4.9	96.8	8.0	19.6		
PMP -75		35.6	32.5	5.4	100				
PRC -.2		23.6	26.7	0.4	5.8	14.3	5.2	21.3	5.3
PRC -.3		22.3	16.3	0.7	12.9	25.9	7.1		
PRC -.4		33.7	10.0	1.0	25.5	34.5	7.5	18.9	20.9
PRC -.5		31.1	8.7	1.2	29.3				
PRC -.6		33.2	11.9	1.5	39.9	9.0	8.0	2.5	0.9
PRC -.7		33.4	9.4	2.2	68.9				
PDMST		29.3	7.0	1.4	36.7	20.7	6.2	13.5	1.7
PDMSTA		16.0	8.1	1.2	21.7	6.7	4.5	18.5	2.2
PPESS - 1		24.4	2.8	2.5	67.9	19.4	1.7		
PPESS - 2		16.6	1.8	2.7	85.7				
PPESS - 5		7.9	1.0	3.2	100	5.2	0		
PPGV	*	64.5	9.8						
PMLGV		17.7	12.7	4.5	95.3				
SSPGV		23.9	9.3		0				
PSD		25.6	1.8	2.1	68.6	35.2	0.3		
PSDML		13.0	8.9						
PMLSD		45.0	6.5						

key:

!	* indicates that oscillations were encountered
D	average disutility
I	average number of iterations to convergence
P1	average number of nodes updating per iteration
P2	percent of iterations with a parallel update
VDP	variability in D due to prime
VIP	variability in I due to prime
VDS	variability in D due to network size
VIS	variability in I due to network size

Table 2

NAME		UNCERTAINTY						
		GUF	GUM	LVUF	LVUM	SUUF1	SUUF2	SUUM
SSP		4.0	14.0	4.0	14.0	0	0	0
PP		48.5	37.0	10.8	13.3	89.7	89.0	12.2
PML PMLA		28.7	18.0	33.2	7.7	96.2	95.5	2.6
	-25	40.8	20.7	34.4	8.0	97.1		2.5
	-40	40.2	22.1	44.1	8.1	96.5	94.6	3.6
PMP	-50							
	-60							
	-75							
	-.2							
	-.3	13.9	19.7	13.5	18.3	85.7	11.1	10.2
PRC	-.4	19.0	27.7	15.5	18.7	90.2	23.0	10.1
	-.5	14.9	22.1	13.8	16.5	90.2	26.4	8.8
	-.6	24.0	25.1	16.8	18.1	93.7	37.2	10.1
	-.7							
PDMST		20.9	14.3	26.6	18.3	62.8	23.0	7.8
PDMSTA		18.0	18.6	19.3	16.5	65.7	14.3	6.6
	- 1	8.9	10.6	17.9	13.6	100	67.9	10.2
PPESS	- 2	17.4	20.0	8.6	13.3	60	51.4	10.0
	- 5	0	0	0	0	0	0	0

uncertainties:

GUF - Gross Uncertainty Frequency
GUM - Gross Uncertainty Magnitude
LVUF - Local-View Uncertainty Frequency
LVUM - Local-View Uncertainty Magnitude
SUUF1 - Simultaneous-Update Uncertainty Frequency for iterations with parallel updates
SUUF2 - Simultaneous-Update Uncertainty Frequency for all iterations
SUUM - Simultaneous-Update Uncertainty Magnitude

Table 3

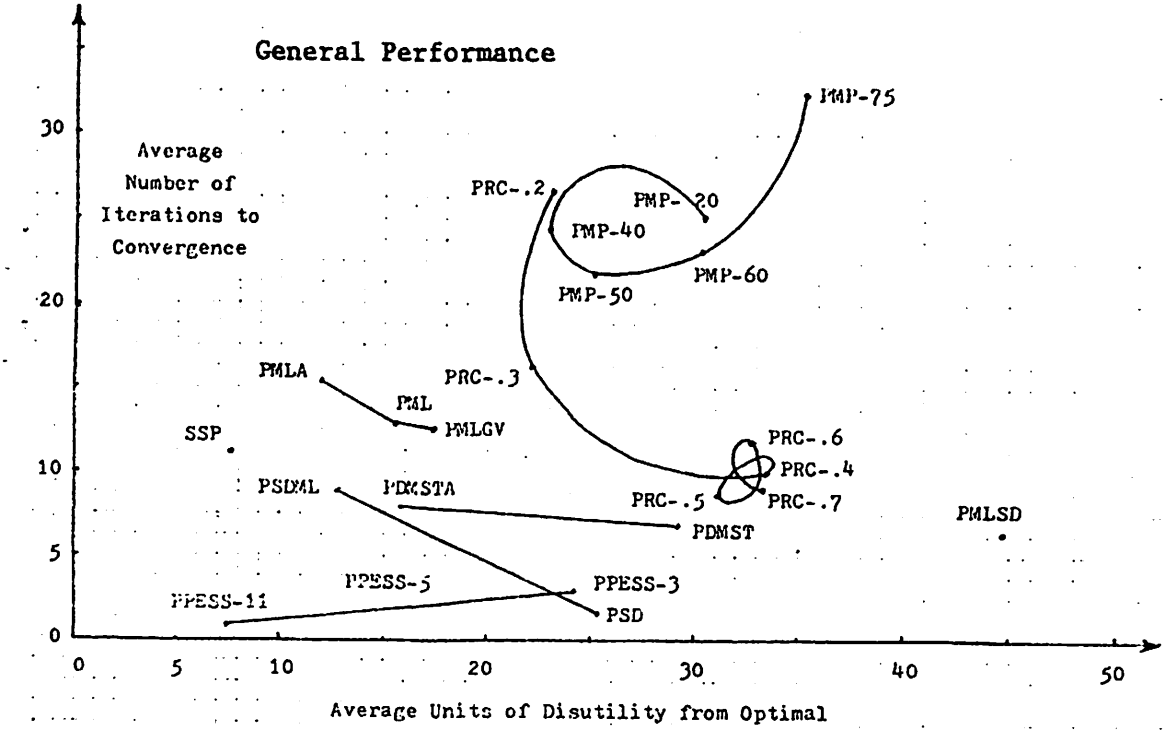


Figure 7

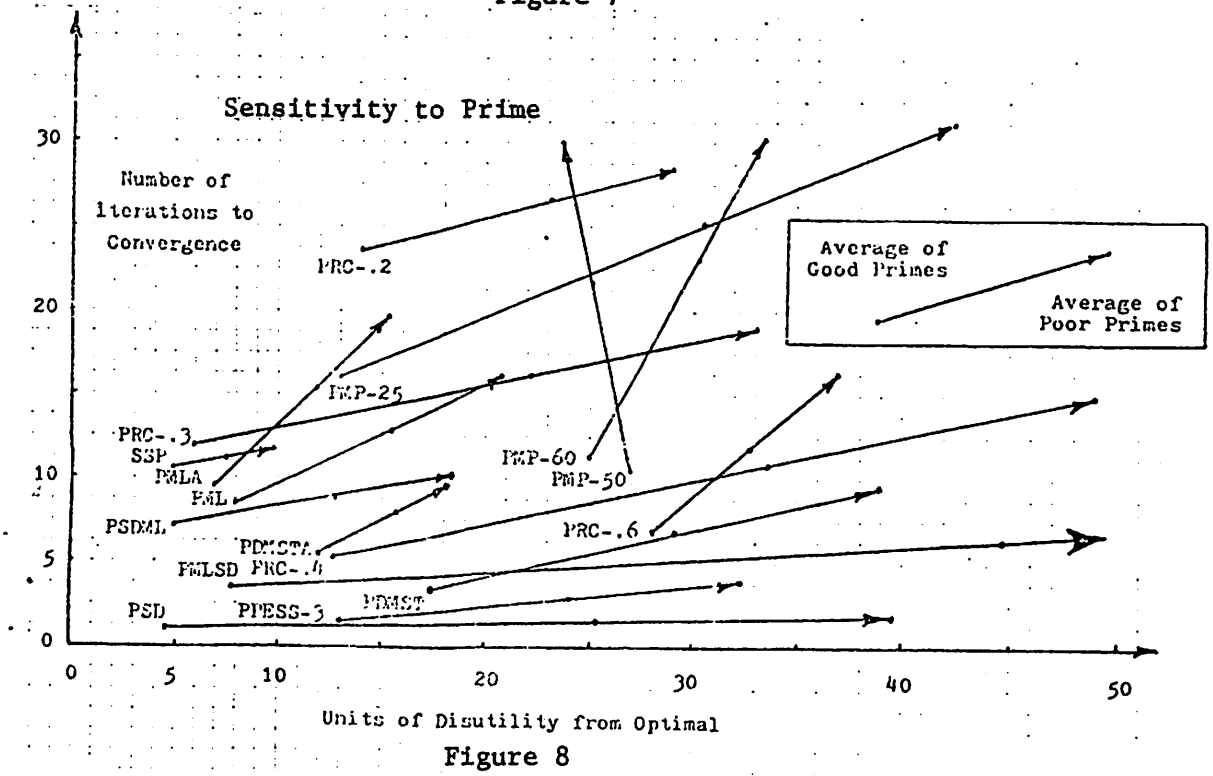


Figure 8

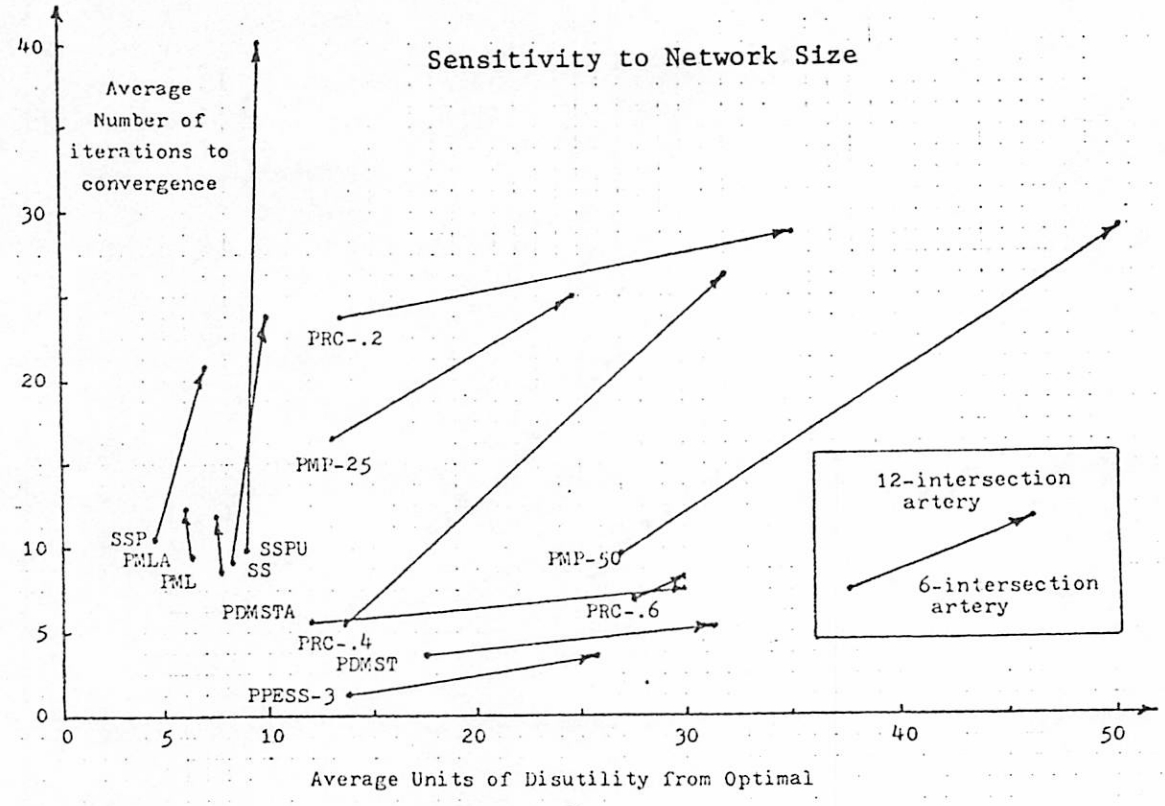


Figure 9

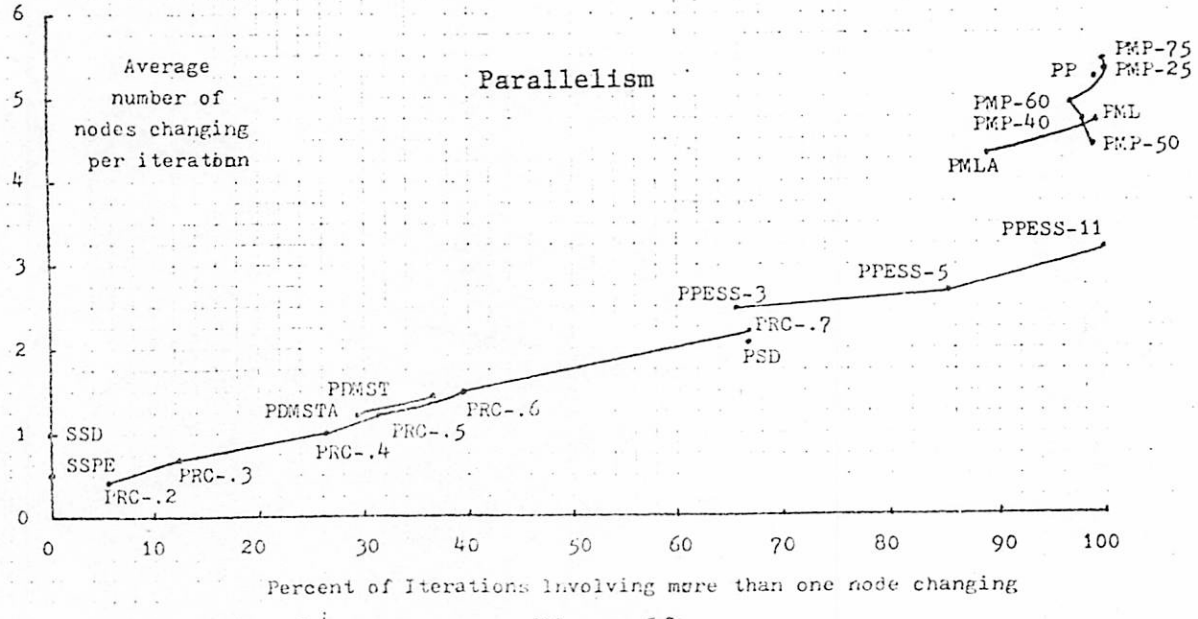
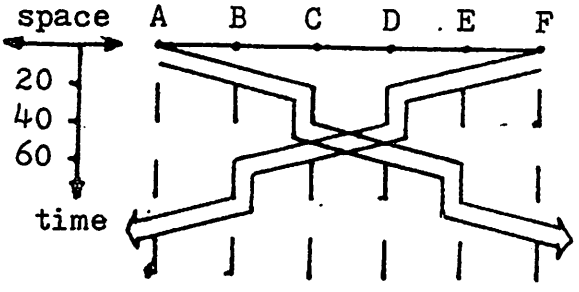


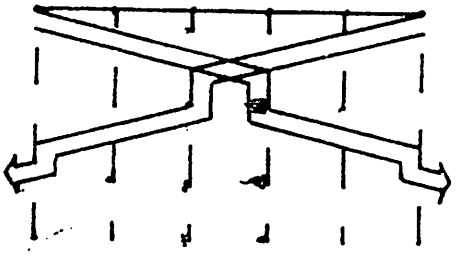
Figure 10

FREE-FLOW DELAYS: AB=10, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT=10, RIGHT=10
****

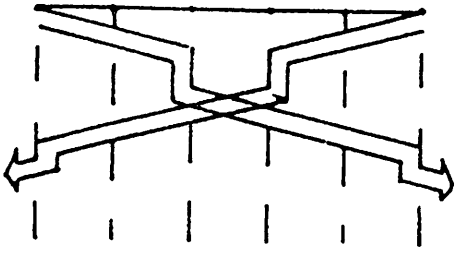
ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	0	0	0	0	0	0	176
1	0	10	10	10	10	0	144
2	0	10	0	0	10	0	144
3	0	10	10	10	10	0	144
4	0	10	0	0	10	0	144
.							
.							
.							



I=0



I=1



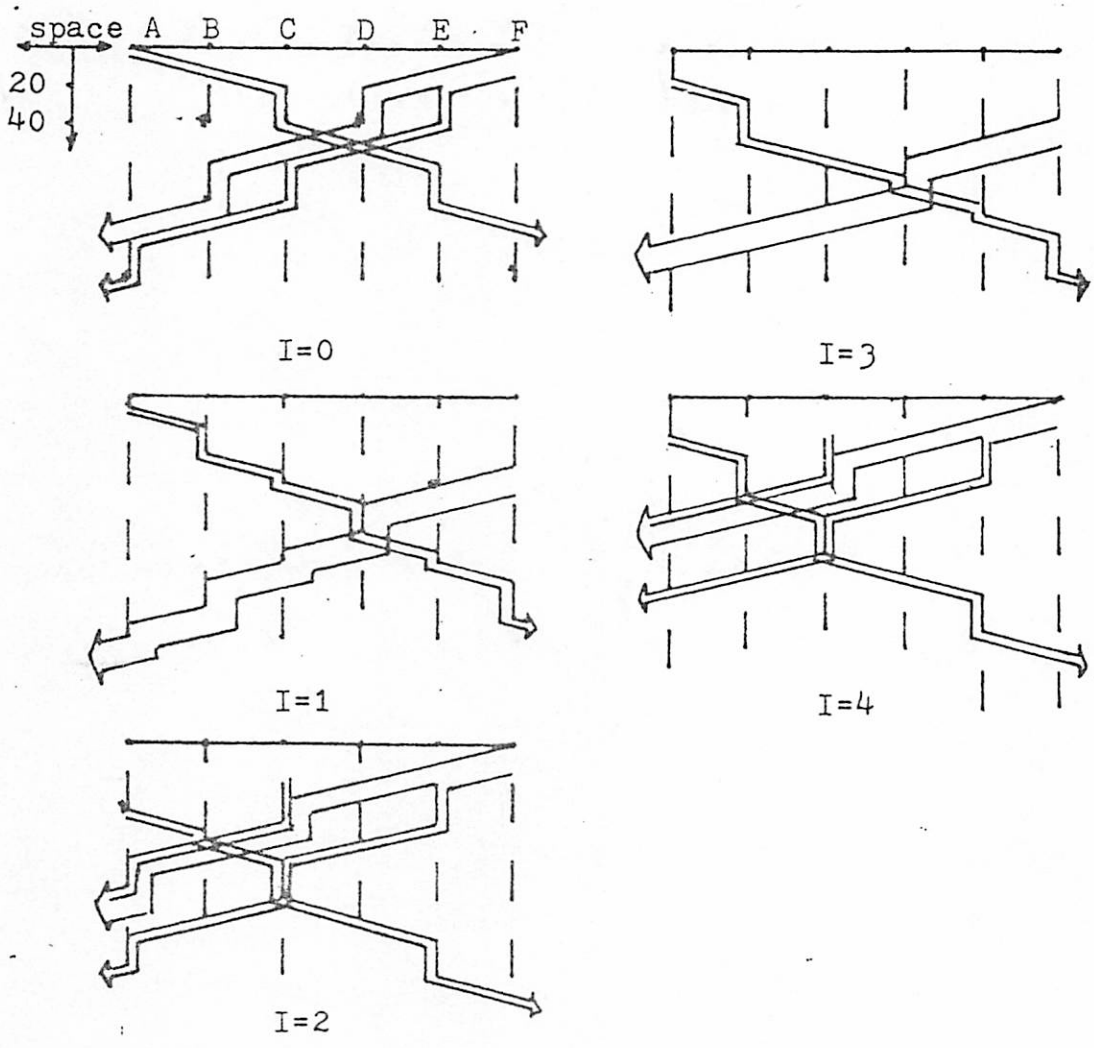
I=2

Run of PP, the Pure Parallel Version
Figure 11a

FREE-FLOW DELAYS: AB=10, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT=15, RIGHT= 5
****

ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	0	0	0	0	0	0	242
1	5	30	5	30	5	35	246
2	35	10	0	10	0	0	194
3	15	5	0	30	5	35	118
4	20	10	0	10	0	0	152
5	15	5	0	30	5	35	118
6	20	10	0	10	0	0	152

.
.
.



Run of PP, the Pure Parallel Version
Figure 11b

FREE-FLOW DELAYS: AB=10, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT= 5, RIGHT=15
****

ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	0	10	20	30	0	10	110
1	5	5	15	25	35	5	198
2	0	10	20	30	0	15	136
3	5	5	15	25	10	5	184
4	0	10	20	0	0	15	126
5	5	5	15	25	10	20	172
6	0	10	20	5	15	15	144
7	5	5	15	25	10	20	172
8	0	10	20	5	15	15	144
.							
.							
.							

Run of PP, the Pure Parallel Version
Figure 11c

FREE-FLOW DELAYS: AB=20, BC=10, CD=10, DE=10, EF=10
PLATOON BANDWIDTHS: LEFT=10, RIGHT=10
****

ITERATION	SG_A	SG_B	SG_C	SG_D	SG_E	SG_F	DISUTILITY
0	0	0	0	0	0	0	264
1	20	10	0	10	0	20	216
2	30	0	0	0	20	0	200
3	20	10	10	0	10	20	188
4	0	0	10	0	0	10	204
5	20	20	10	10	10	20	264
6	0	0	20	0	20	10	300
7	20	20	0	10	10	20	216
8	0	0	20	10	20	0	288
9	20	20	10	10	10	20	264
.							
.							
.							

Run of PP, the Pure Parallel Version
Figure 11d

is assumed to result in a global increase in goodness. Second, neighboring subproblem solutions are assumed to be correct and to remain constant during the time a node makes its update.

The first of these assumptions is often invalid, especially where there is strong interaction between non-neighboring subproblems. Thus, uncertainty which we call local-view uncertainty may be present. Figure 6 illustrates how local-view uncertainty can lead to the possibility that an update will decrease global goodness.

Local-view uncertainty can affect the stability of an IR algorithm since refinements of a subproblem may lead to a decrease in global goodness and can undo previous refinements of interdependent subproblems, leading to possible oscillatory behavior. Note that local-view uncertainty is present in SIGOP II, and as a result, convergence of the SIGOP II algorithm cannot be guaranteed.

The second assumption, that neighbors will not alter their control and state variables, is violated by parallel updates, thus introducing simultaneous-update uncertainty into the computation. This type of uncertainty is not present in SIGOP II since there are no parallel updates. Another type of uncertainty can be introduced if the platoon-flow in the network is not completely recalculated after each local refinement. We call this type of uncertainty environmental-update uncertainty, and in addition to simultaneous-update uncertainty and local-view uncertainty, it too can affect stability. These types of uncertainty are introduced by an algorithm's control structure which governs the way (local) updates are combined.

Other assumptions lead to uncertainties which do not affect the stability of single-label IR algorithms. For example, an initial assumption of a prime determines where in the overall search space the search will start, and thus may affect the outcome of the search, but does not affect an algorithm's ability to converge. The assumption that the global metric (in our case, disutility) indicates some measure of distance from the optimal solution leads to uncertainty when multiple local (non-optimal) maxima exist using the metric. The uncertainty concerns whether an update that increases the global metric will lead the algorithm towards the optimal solution or towards a non-optimal solution. Depending upon the shape of the disutility "surface", it may be possible for a move towards a non-optimal solution (and away from the optimal solution) to increase the global metric. There is little that can be done to correct this problem because there is no way to distinguish between the arrival at an optimal or non-optimal fixed-point (i.e., an overall solution that is locally coherent in that further refinements are not possible).

4.1. Distributed SIGOP II Test Versions

A number of test algorithms were developed which are based on the same traffic flow model and local optimizations as SIGOP II, but which perform local optimizations in a lock-step parallel fashion. Section 4.1.1 describes and reports results of one of these parallel algorithms that employs no inter-node coordination. The remainder of Section 4.1 concerns a number of versions which employ various inter-node coordination schemes to control or compensate for the uncertainties observed in the pure parallel version. These coordination schemes include modulation of updates, reduction of parallelism, and extension of local neighborhoods. Several additional versions are introduced in Section 4.2 for comparison, and a general discussion of single-label IR follows in Section 4.3. Results of all these versions are tabulated in Tables 1-3, and are graphically presented in Figures 7-10.

4.1.1. Pure Parallel Version

The first parallel test algorithm, PP, uses the same traffic flow model and local optimizations as SIGOP II, but performs local optimizations in a lock-step parallel fashion with no other inter-node coordination. After each iteration, the platoon structure is updated and the tentative solution is evaluated as is done after sweeps in SIGOP II. It is hoped that the need for lock-step synchronization and complete platoon structure updating between iterations (which provides a consistent base for evaluation and further updates) can be relaxed in later versions. An alternative approach which does not require this global calculation will be discussed in Section 5.

Sample runs of the algorithm are depicted in Figures 11a-d. Since SIGOP II's priming process is serial in nature and thus not suited to PP, performance is based on twenty test cases which consist of the same 6, six-intersection arterial configurations used to test SS but with a number of primes. The primes for each of the six configurations include arbitrarilly chosen poor primes, and good primes which represent near-optimal solutions.

The results indicate that PP's performance is quite poor. The quality of solution, which is inversely related to the difference in disutility between the solution and optimal, was quite low despite the fact that the best tentative solution of a run was taken to be the final solution. The optimal solution was found in only one case out of twenty. Convergence took place only in this one case where an optimal solution was found. Instead of convergence, large oscillations occured; these oscillations

span from one to seventeen (8 on average) iterations. Detection of such oscillations requires considerable memory and processing time.

A number of measurements of various types of uncertainties present in the PP algorithm were taken. The frequency and average magnitude of error in the assumption underlying each uncertainty was measured. Local-view uncertainty is measured by comparing the disutility associated with the current network timing plans to the disutility associated with network timing plans generated by changing only a single signal timing plan. Simultaneous-update uncertainty was measured by comparing the assumed values of neighbors' timing plans and platoon structure before updates with the actual values that result after updates.

An additional measure of uncertainty, gross uncertainty, is a combination of the other uncertainties. Gross uncertainty indicates overall uncertainty as to whether an iteration's updates will decrease global disutility or not. Note that an algorithm with no gross uncertainty will always converge since global disutility will always be decreasing. For PP, 47% (almost half) of iterations result in an increase in global disutility; the magnitude of the increases was on average 37 units of disutility. This is in comparison to version SSP where the frequency of error due to gross uncertainty was found to be 4%, and the magnitude of error averaged 14 units of disutility.*

Frequency of errors in predicting the worth of changes to signal timing plans due to local-view uncertainty, measured as the percent of iterations where one or more nodes make updates based on local views that alone would increase global disutility, was found to be 13%; the average magnitude of error in the predicted worth of changes averaged 13 units of disutility. Simultaneous-update uncertainty was also measured; 90% of iterations involving a simultaneous-update had some information about the value of a neighbor's state variables (platoon departure times) or control variable (signal switching time) invalidated by the parallel update. The average magnitude of error in either control or state variables due to simultaneous-update uncertainty averaged 12 seconds. Note that the error could be at most half the cycle length (in this case 20 seconds).

---

* Note that the disutility value of network timing plans varied from 0-28 units of disutility for optimal network timing plans, to 200-350 units of disutility for poor timing plans.

span from one to seventeen (8 on average) iterations. Detection of such oscillations requires considerable memory and processing time.

A number of measurements of various types of uncertainties present in the PP algorithm were taken. The frequency and average magnitude of error in the assumption underlying each uncertainty was measured. Local-view uncertainty is measured by comparing the disutility associated with the current network timing plans to the disutility associated with network timing plans generated by changing only a single signal timing plan. Simultaneous-update uncertainty was measured by comparing the assumed values of neighbors' timing plans and platoon structure before updates with the actual values that result after updates.

An additional measure of uncertainty, gross uncertainty, is a combination of the other uncertainties. Gross uncertainty indicates overall uncertainty as to whether an iteration's updates will decrease global disutility or not. Note that an algorithm with no gross uncertainty will always converge since global disutility will always be decreasing. For PP, 47% (almost half) of iterations result in an increase in global disutility; the magnitude of the increases was on average 37 units of disutility. This is in comparison to version SSP where the frequency of error due to gross uncertainty was found to be 4%, and the magnitude of error averaged 14 units of disutility.*

Frequency of errors in predicting the worth of changes to signal timing plans due to local-view uncertainty, measured as the percent of iterations where one or more nodes make updates based on local views that alone would increase global disutility, was found to be 13%; the average magnitude of error in the predicted worth of changes averaged 13 units of disutility. Simultaneous-update uncertainty was also measured; 90% of iterations involving a simultaneous-update had some information about the value of a neighbor's state variables (platoon departure times) or control variable (signal switching time) invalidated by the parallel update. The average magnitude of error in either control or state variables due to simultaneous-update uncertainty averaged 12 seconds. Note that the error could be at most half the cycle length (in this case 20 seconds).

---

* Note that the disutility value of network timing plans varied from 0-28 units of disutility for optimal network timing plans, to 200-350 units of disutility for poor timing plans.

4.1.2. Modulated Versions

The first attempt to improve PP, the pure parallel version, involved modulating (reducing) the magnitude of updates. The new version, PML, reduces the magnitude of updates in the following way: a desired change in green switching time of more than 5 seconds is limited to 5 seconds, and a desired change less than or equal to 5 seconds is limited to 1 second. A number of improvements should be realized in the new algorithm including a reduction of the magnitudes of simultaneous-update and local-view uncertainties because changes in timing plans are limited, and a capability to gather additional information from the extra searches made along the projected way. Note that modulation is a purely local mechanism, and hence is ideal for our distributed problem solving environment.

The performance of PML was measured utilizing the same twenty test cases used to test PP, and was in general quite good. On the average, solutions obtained using PML had about 50% more excessive disutility* than solutions obtained using SSP, the best SIGOP II test version. Convergence for PML took about 20% more iterations than SSP (refer to Table 2 and Figure 7). Recall that a single iteration for PML involves N updates in parallel (where N is the number of nodes), whereas a single iteration for SSP involves only one update; although a single iteration for PML involves N times the computation required for a single iteration of SSP, the time it takes for both algorithms to advance an iteration is the same.

Although a small oscillation around a solution was often encountered, it spanned only two iterations and was therefore as easy to detect as normal convergence. For this reason, convergence was considered to have been achieved when either true convergence (stability) or small oscillation was detected. PMLs sensitivity to prime is 2.5 times that of SSP (see Figure 8).

A very strong point for PML is its low sensitivity to network size (see Table 2 and Figure 9). The quality of solutions is affected very little by a doubling of the network size and the number of iterations required for convergence increased by only a third. SSP (and SS), on the other hand takes twice as many iterations when the network size is doubled. These results indicate that PML should be significantly faster than SS on very large networks.

PML, as did PP, exhibits a high degree of parallelism, which accounts for its high speed of convergence despite the reduction in magnitude of changes. Measurements of the

---

* Excessive disutility is disutility in excess of that found in an optimal solution for the configuration.

uncertainties in PML confirm expectations that the magnitudes of uncertainties would be reduced through modulation. The magnitudes of gross and local-view uncertainties was reduced by 40-50% (compared to PP), and that of simultaneous-update uncertainty was reduced by 78%!

A small improvement in the quality of solutions was achieved with PMLA, a modified PML, at slight additional cost to speed of convergence. PMLA incorporates a mechanism which breaks small oscillations to allow convergence to the nearby solution. Since this mechanism only comes into play if a small oscillation is detected, the behavior of PMLA is very much like PML (see Table 2 and Figures 7-10).

Tests on versions PMP-25, PMP-40, PMP-50, PMP-60, and PMP-75 which employ modulation of updates by a percentage indicate that modulation by a percentage is an inferior mechanism for controlling uncertainty when compared to modulation by a fixed limit as employed by PML and PMLA (see Tables 2 and 3, and Figures 7-10) Not only was the quality of solution and speed of convergence poorer, oscillation spanning multiple iterations was observed in PMP-25, PMP-40, and PMP-50.

4.1.3. Reduced Parallelism Versions

Another approach to reducing simultaneous-update uncertainty is to reduce the potential for conflicting interaction among nodes by reducing parallelism. One way to accomplish this is to have each processor draw a random number before each iteration to determine if it is allowed to change. Test versions PRC-.2 through PRC-.7 employ this strategy with a probability of a node being allowed to change on a given iteration of 0.2 through 0.7, respectively. This rule is well suited for distributed processing because it involves no global coordination.

As might be expected, a reduction of the frequency of simultaneous-update uncertainty was realized. The reduction was indirectly related to the probability of a node being allowed to change, and also led to a decrease in gross uncertainty frequency. Unfortunately, however, performance was mediocre at best and significant parallelism was lost (see Tables 2 and 3, and Figures 7-10).

An interesting observation is that although the quality of solutions is poor when a node's probability of changing is moderate or high, speed of convergence is reasonable and no large oscillations were encountered. This was apparently due to randomness of updates which prevents a reoccurrence of the same sequence of updates that comprised the oscillation. On the other hand, this randomness in updating

was also responsible for slow convergence when the probability of updating was low.

A second method of reducing conflicting node interaction, and hence reducing simultaneous-update uncertainty, involves incorporating a version of the maximal spanning tree (MST) heuristic in the parallel algorithm. SIGOP II utilizes an MST to order updates. A new parallel algorithm uses the MST to give priority to some updates by inhibiting others. As with SIGOP II, the MST is determined once, on the basis of current link traffic volumes, before solving for a network timing plan.* Once the maximal spanning tree has been established, a local rule can be used to determine when nodes may make updates.

A new version, PDMST, employs such a rule: an update for a node is allowed if the node's neighbors which are higher (closer to the "root") on the MST do not wish to change. Performance of the PDMST algorithm was in many ways similar to the performance of PRC-.4, from quality of solution and degree of parallelism, to the magnitudes and frequencies of uncertainties (see Tables 2 and 3). Note that simultaneous-update uncertainty between neighbors involving control variables (signal switching times) has been eliminated in PDMST because settings of neighboring signals cannot both be altered. However, simultaneous-update uncertainty between neighbors involving state variables (platoon departure times) remains a problem due to non-local subproblem interaction.

Another version, PDMSTA, utilizes a more complex MST heuristic. This new heuristic incorporates additional features found in SIGOP II's MST heuristic. For example, the direction of priority along the MST is reversed after each node has had a chance to update (as was the sweep direction for SIGOP II), and nodes are not permitted to update more than once until the priority direction changes. As might be expected, the parallelism of PDMSTA is less than that of PDMST. However, the quality of solutions was significantly improved (see Tables 2 and 3, and Figures 7-10) with only a slight increase in the number of iterations to convergence.

The shortcoming of PDMSTA appears to be the algorithm's sensitivity to network size. The quality of solutions was decreased 2.5 times with a doubling of the network size. This is apparently so because the distributed MST heuristics do not really coordinate changes of non-neighboring signals, and an increase in interaction among signals appears to accompany an increase in the number of signals in the

---

* See [Dalal 1977] and [Spira 1977] for distributed algorithms for determining the maximal spanning tree of a network.

network.

4.1.4. Extended Neighborhood Versions

A final attempt to modify the pure parallel test version PP so as to reduce simultaneous-update uncertainty involved replacing the assumption that control and state variables of a node undergoing an update will not change with a prediction of their values. In order to make accurate predictions for its neighbors' actions, a node must plan for its neighbors. This requires that the node communicate with its neighbors' neighbors. The cost of these predictions is an increase in the size of local search spaces, accompanied by an increase in communication required to support the extended neighborhoods.

For an arterial network, where each node has at most two neighbors to plan for, the local search space size is cubed. Because a larger search space will take longer to search an approximate search technique is appropriate. PPESS is a new algorithm that employs a small (serial) SIGOP II search on each extended neighborhood to plan for neighbors' actions and determine updates. This embedding of the SIGOP II heuristic search into the update computation replaces the complete local search performed by each node. Version PPESS-1 makes predictions for neighbors, and PPESS-2 makes predictions for neighbors and neighbors' neighbors. In general, version PPESS-x involves making predictions for x nodes on each side of the node.

Test results of PPESS are included in Tables 2 and 3, and Figures 7-10. A definite improvement over PP is realized by the PPESS versions. The quality of solutions found by PPESS-1 was mediocre, but convergence was fast and no oscillations were encountered. Further improvement can be realized by increasing the neighborhood size. However, communication costs, as well as time/space requirements increase as the neighborhood size increases.

Variation in the quality of solutions found by PPESS-1 given various primes was moderate; variation in speed of convergence was low. Surprisingly, measurements of error due to the uncertainties in PPESS-1 indicate that although both the frequency and magnitudes of error due to local-view and simultaneous-update uncertainties was moderate, the frequency and magnitude of gross uncertainty error was low. This discrepency can perhaps be explained by noting that the measures of error due to local-view and simultaneous-update uncertainties indicate the percentage of iterations where at least one instance among all nodes is detected rather than how often a single node makes an error.

4.2. Additional Parallel Versions for Comparison

4.2.1. Global View Versions

In order to determine the extent to which local-view uncertainty affects performance, comparison versions PPGV, PMLGV, and SSPGV were developed. In previous versions, a node's search space, used to determine an updated signal setting, represents a local view of the affects of changes in the node's signal setting,. For the new global view versions, the global rather than local value of changes are computed in constructing a signal's search space. This is done in a straight forward manner, one node at a time, at the expense of significant additional computation and communication. Note that neighbors are still assumed to remain unchanged during nodes' updates. Thus, when parallel updates are allowed, simultaneous-update uncertainty still remains.

Version PPGV is as unstable as version PP. Performance of versions PMLGV and SSPGV (see Tables 2 and 3, and Figures 7-10) indicate that a little local-view uncertainty is, in fact, beneficial. It appears that the algorithms can recover from a small number of "incorrect" refinements, and the incorrect updates may lead to the gathering of additional information, which in turn leads to better solutions.

4.2.2. Global Coordination Versions

A new parallel algorithm, PSD, was developed which utilizes extensive communication and computation to achieve global coordination of parallel updates based only on local information. This version was developed to see if brute force compensation for local-view and simultaneous-update uncertainties would produce a parallel algorithm which performs well. The values of all combinations of updates selected by nodes' local searches are determined, the best combination is selected and implemented, and then the platoon structure is fully updated before the process is repeated. The new algorithm converges very quickly to an optimal solution if primed with a near optimal prime. Rapid convergence to non-optimal solutions was observed, however, when poor primes were used (see Figure 8).

Again, as with SSPA and SSD, the algorithm appears "too stable" to find a good solution when started with a poor prime. It was conjectured that modulation might provide for the aquisition of new information which might lead to better solutions. The version PSD was therefore modified appropriately, yielding PSDML. Results confirm our

expectations of improved quality of solutions (see Figure 7).

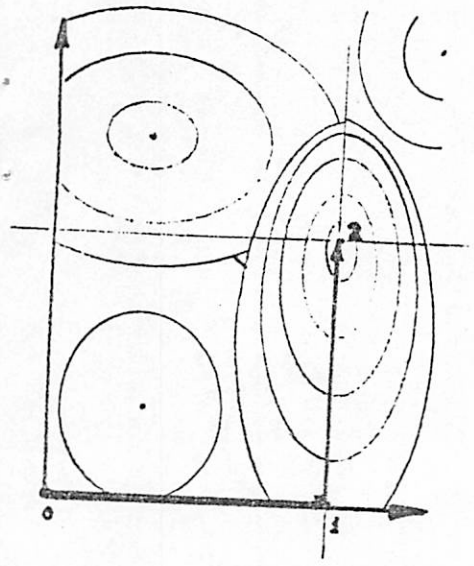
4.3. Discussion

In Section 4.2.1, the results of tests of a straight forward IR algorithm for distributed network traffic light control were reported. This algorithm, PP, failed to converge in most cases because of excessive simultaneous-update uncertainty. A number of modified versions of this algorithm were developed for which the simultaneous-update uncertainty was lessened by various mechanisms as indicated in Table 1. These algorithms include PML and PMLA which employ modulation to reduce the magnitudes of uncertainty, PRC and PDMST which employ local rules to reduce simultaneous-update uncertainty by coordination, and PPESS which utilizes predictions for neighbors' changes to reduce simultaneous-update uncertainty.

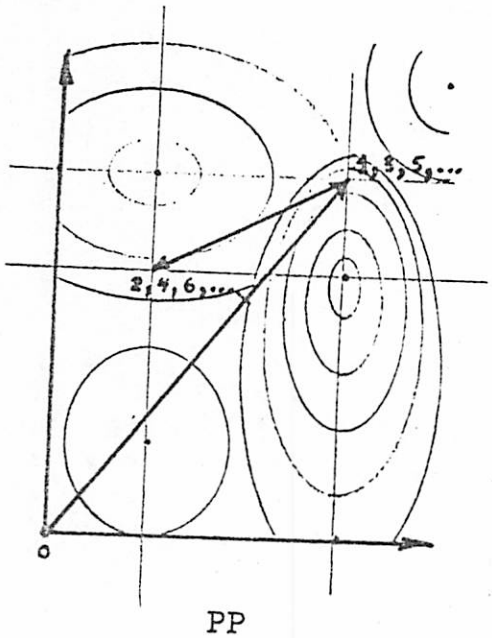
The algorithms were faced with the task of searching a large search space of high dimensionality with multiple maxima/minima and discontinuities. The algorithms are similar in many respects to classical "hill-climbing" algorithms [Cooper and Steinberg 1970]. In fact, the SIGOP II algorithm upon which these distributed algorithms are based, is a univariate algorithm that employs a heuristic ordering for one-dimensional (line) searches.

The use of line searches along each dimension instead of using the gradient to determine updates is highly advantageous when starting with a poor prime (see Figure 12) because the algorithm is less susceptable to convergence to non-optimal solutions. In a distributed environment line searches on the global metric contours are not feasible because of the communication required to determine the global contours. Approximations based only on local information must be substituted, thereby introducing local-view uncertainty into the computation. In addition, strong interaction between dimensions make the values of combined (simultaneous) updates uncertain, even given accurate values for each individual update.

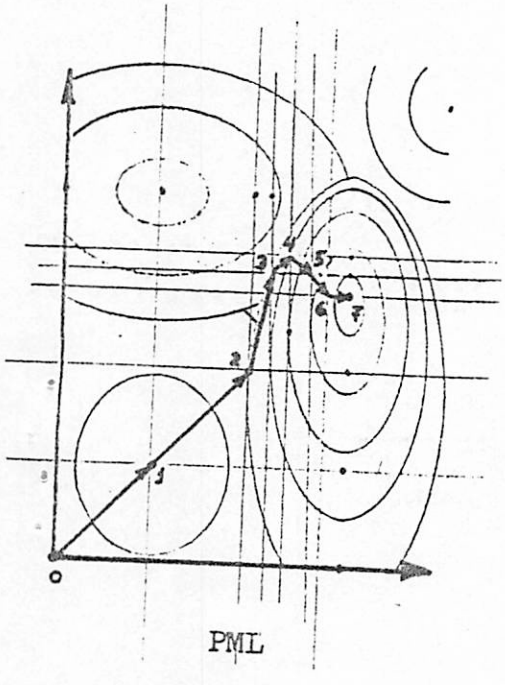
These uncertainties combined with the fact that the magnitude of updates do not decrease as the number of iterations increases, make a guarantee of convergence also impossible. The existance of multiple non-optimal solutions and the lack of a backtracking mechanism or complete search make a guarantee of finding an optimal solution impossible. Despite these limitations, many of the algorithms appear quite robust in finding optimal or nearly optimal solutions with reasonable speed.



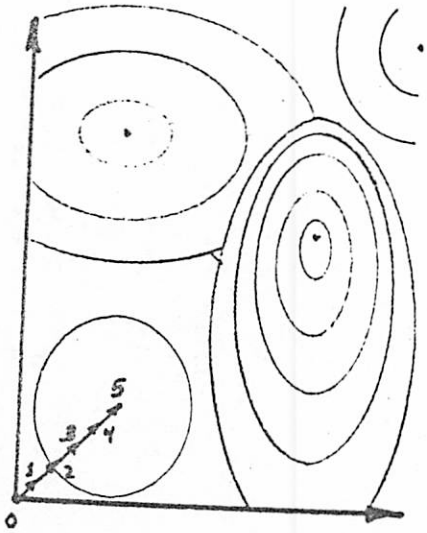
SSP



PP



PML



GRADIENT

Examples of Algorithms on a Two-Dimensional Search Space

Figure 12

The excellent performance of the serial SIGOP II algorithm was difficult to duplicate with a distributed algorithm. SIGOP II's excellent performance appears primarily to be due to its MST heuristic node processing ordering. This heuristic introduces implicit non-local coordination among nodes by controlling the order of node updating. The heuristic can also be viewed as controling the flow of information in the network about changes to timing plans and platoon structures. Such a heuristic is, however, difficult to incorporate in a parallel algorithm without loosing significant parallelism and without using costly global coordination.

PMLA is, perhaps, the most promising of the distributed algorithms. Results of tests indicate that PMLA is almost as good as SIGOP II for small networks and could be faster than SIGOP II for larger networks. Uncertainty in PMLA is controlled by a reduction in the magnitudes of changes (modulation), which causes a reduction in possible error, and hence in uncertainty.

From a distributed processing viewpoint, the need for lock-step synchronization and complete platoon structure updating between iterations in the parallel, single-label IR algorithms developed is highly problematic. Complete platoon structure updating appears necessary for the accurate evaluation of a potential solution, as well as for the detection of proper convergence. It is possible that incomplete environmental updating with periodic complete environmental updating will work; however, experiments with partial platoon structure updating have not as yet been tried.

Before moving on to a multi-label IR approach to the NTLC problem, it should be noted that for a number of other problem domains, such as image processing and possibly air traffic control, non-local dependencies are fewer and/or weaker. Often when this is the case, changes to labels affect only nearby nodes. This means that local-view uncertainty is lessened because the local effects of a change in a label accounts for most of the overall effects of the label change. Furthermore, simultaneous-update uncertainty is lessened because the probability of interacting non-neighboring nodes is lower, and partial environmental updating between iterations might be sufficient to obtain a consistent environment for further updates.

5. A Multi-label Iterative Refinement (RELAXATION) Approach

Our second approach towards developing a distributed algorithm for determining optimal network timing plans involves using a multi-label iterative refinement formulation. In this formulation a node holds a probability distribution of its possible partial solutions rather than a single partial solution. Algorithms based on this formulation are called multi-label IR algorithms because all possible alternatives (the labels) are utilized.

By holding alternative timing plans and manipulating the confidence values of the alternatives, simultaneous-update uncertainty as found in the single-label IR algorithms can be reduced. This is because changes in confidence values are kept small (except when a label is totally incompatible) and thus do not eliminate alternatives some neighboring node might assume to exist. Note the similarity of this mechanism to modulation as used in the single-label algorithm, PML. Modulation in PML limits changes in labels (only one is held at any one time) to control simultaneous-update uncertainty.

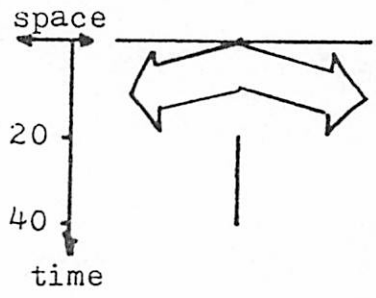
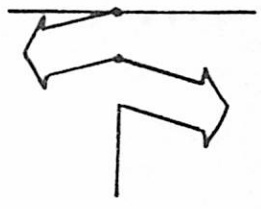
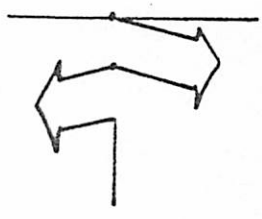
The first complication encountered with this new approach involves updating the platoon structure in response to changes in signal timing plans. The need for updating the platoon structure can be eliminated by introducing a platoon structure component into timing plans. Thus, a signal timing plan consists of a cycle length, green and red switching times, and platoon departure times for each direction (see figure 13).

A multi-label IR algorithm adjusts the confidence values of alternatives until further adjustments are sufficiently small. At this point it is hoped that at each signal the confidence values are one for the alternative which is part of the optimal global solution and zero for the other alternatives. Initial confidence values can be equal (corresponding to a no-information state) or not equal, allowing for a heuristic prime. The adjustments weigh the confidence values of alternatives, and the compatability associated with the combinations of alternatives. These compatabilities are determined on the basis of platoon structure consistency and disutility.

This new multi-label iterative refinement algorithms is modelled after the "relaxation" algorithms used in a number of AI applications, paritcularly those involving machine vision.* Some of these applications include: image segmentation [Hanson and Riseman 1978a], vertex labeling in blocks-wourld scenes [Waltz 1975], line and curve

---

* The relaxation technique was originally developed for the solution of simultaneous differential equations.

LABEL	TIMING PLAN* sg	sr	td_{left}	td_{right}	PICTURE
λ_1	0	20	0	0	
λ_2	0	20	0	10	
λ_3	0	20	10	0	
. . .					

* common cycle length = 40 sec.

Labels composed of
signal switching times and platoon departure times

Figure 13

enhancement [Zucker, Hummel, and Rosenfeld 1975], spring-loaded template matching [Davis and Rosenfeld 1976], interpretation guided segmentation [Tenenbaum and Barrow 1976], and hierarchical waveform parsing [Davis and Rosenfeld 1977]. In these applications, confidence values associated with labels, representing alternative interpretations of objects, are updated in accordance with the confidence values of an object's own and neighboring objects' labels, and compatibility relations between labels of objects. A relaxation process has also been used as a model of the vertebrate central command system [Kilmer et.al. 1969].

It should be noted that all signals do not necessarilly have the same set of alternative timing plans (labels). For example, a signal at the left end of an artery will not have labels for which right-going platoons depart midway through the green phase, such as label 2 in figure 13. This is because traffic is assumed to approach the artery from the left at a steady rate which results in the formation of platoons at the left-most signal that enter the artery always at the start of the signal's green phase.

Another complication, equivalence classes for network timing plans, results in the existence of multiple, equivalent solutions. Other solutions within an equivalence class can be obtained from a given solution through a global shift in time of the signals' switching and platoon departure times. A restriction in the set of allowable timing plans at one intersection eliminates the multiple, equivalent plans. A prime of unequal confidence values at intersections provides another means to bias the algorithm towards a unique solution within the the equivalence classes. It should be noted that multiple solutions in different equivalence classes are possible.

An interesting difference between the AI applications which use relaxation and the NTLC application examined here is that the compatibility relation does not change for subsequent runs on different data in the AI applications, whereas a dynamic compatibility relation must be used for the NTLC application. The need for a dynamic compatibility relation is due to the fact that the compatibilities between neighbors' timing plans relates inversly with disutility incurred on the links. Disutility varies for different traffic flow patterns and volumes as well as for different signal settings. Subsequent applications of the relaxation algorithm under changing traffic conditions will require different compatibilities. Thus, environmental data is factored into the compatability relationships rather than into the initial distribution of confidence values for labels as is done in AI uses of relaxation. Although the AI applications have not utilized dynamic compatibilities, the idea could be of some use. For example, image processing applications might benefit from compatibilities which vary

according to the lighting conditions or orientation.

A complication which has been put off for now, but one which will have to be faced before the relaxation approach can be utilized for distributed network traffic light control, is the large number of labels needed to achieve adequate resolution in solving for an optimal network timing plan. For example, a two-dimensional network with an 80 second cycle length and a fixed split of 0.5 requires more than 65,000 labels at each signal to achieve 5 second resolution because all legal combinations of signal switching time and platoon departure times must be represented. More than 4 billion multiplications and additions might be required to update a signal's labels' confidence values! Some ideas for coping with the large label set include: 1) the use of class labels to represent groups of similar and as yet undifferentiated labels; 2) the use of generator functions and thresholding to reduce the number of labels communicated; and 3) the use of processing at multiple levels of resolution to limit the number of labels in use at any one time (a solution at one resolution is used to select labels at higher resolutions to try).

Preliminary tests on arteries with a 40 second common cycle length and 5 second resolution (to limit the number of labels needed at each node to 128) are presented in the next section. Care has been taken to choose test cases which should not need more resolution than is provided. Note that the new algorithm can use the same platoon model and disutility calculations as the single-label SIGOP II versions described earlier in this paper to determine label compatibilities.

5.1. Description of Test Versions and Results

The first step towards developing multi-label test versions was the selection of an appropriate label confidence update rule. The update rule is used once each iteration for each label at each intersection. The updated confidence value for a label at an intersection is a function of the former confidence value of the label, the confidence values of labels at neighboring intersections, and the compatibility between the label to be updated and neighbors' labels.

Two standard update rules are the arithmetic and product rules [Zucker and Mohammed 1978]. These rules along with an additional rule are presented in figure 14. For the arithmetic rule, contributions from neighbors, which are sums of contributions from each label of the neighbor, are summed. The probabalistic product rule multiplies the

contributions from neighbors. The product rule approximates the behavior of the arithmetic rule, but yields faster convergence since the incompatibility of a label with all of some neighbor's labels will quickly decrease the confidence of the label.

The probabalistic product rule presented in Figure 14 sums contributions from a neighbor's labels because a label that is compatible with many labels at a neighboring node has greater probability of being a part of the optimal solution. A product rule with a "fuzzy" set-theoretic rather than probalistic interpretation is also presented in Figure 14. For this rule, the contributions from labels of a neighboring signal are not summed, but rather the maximum contribution by a single label is taken as the contribution of the neighbor.

The compatibility between pairs of labels at adjacent intersections can be determined in a number of ways. Some experiments were carried out with compatibilities which were linearly, but inversely, related to the disutility associated with the use of the timing plans represented by the labels. Other experiments utilized a negative exponential relation between disutility and compatibility. Note that two labels are considered totally incompatible if the platoon structure components of the labels do not fit together. With either method for determining compatibilities, the compatibilities range from a value of zero for totally incompatible labels, to a value of one for labels which are compatible with zero associated disutility.

The use of negative-exponentially related compatibilities appears to be more appropriate for use with a product rule for updating label confidence values because multiplication of neighbors' compatibilities results in a combined compatibility that corresponds to the negative exponential of the sum of the disutilities associated with the links to the neighbors. This coincides with the intent to minimize the sum of the disutility incurred on all network links.

Sample runs of a multi-label IR algorithm using the probabalistic product rule and negative-exponentially related compatibilities are shown in Figures 15a and 15b. Note the "no-information" prime used for nodes, and the restriction of the left-signal to a single green switching time to eliminate multiple equivalent solutions that result from a global time shift. Convergence to poor, non-optimal solutions was observed in all test cases; after 6 iterations, labels which made up the solution were usually the most probable labels for nodes.

The reason non-optimal solutions were obtained seems to be that labels which are compatible with many neighbors' labels dominate over other labels which are compatible with

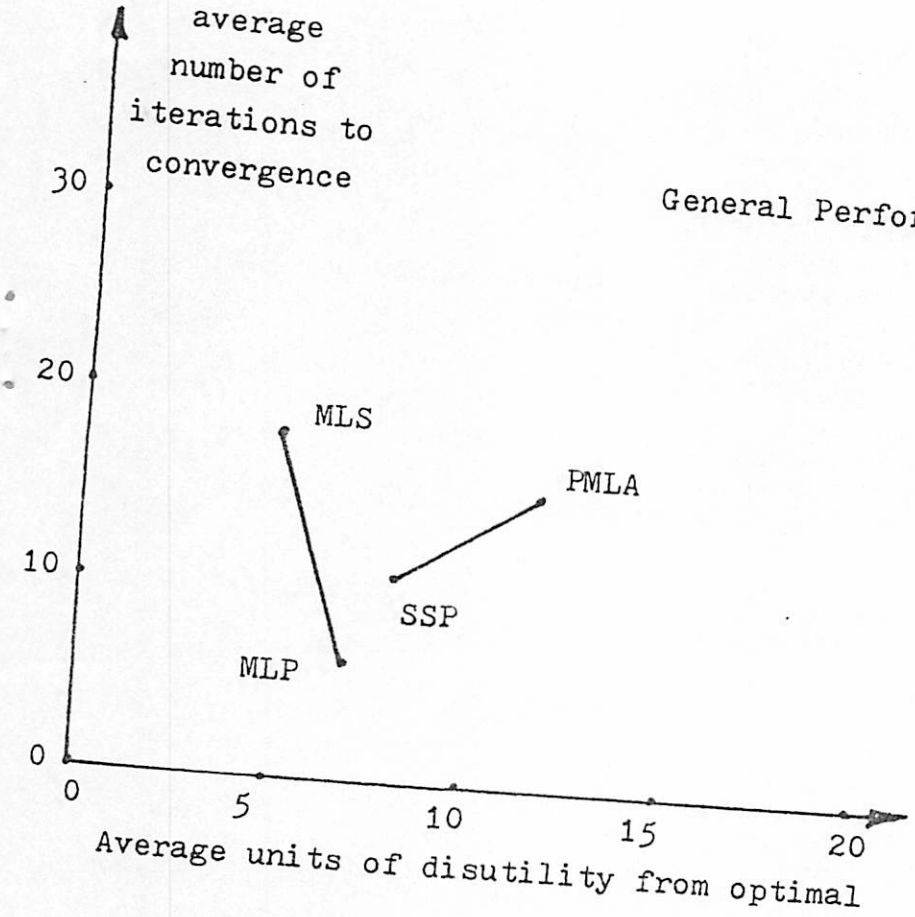
only a few labels. Although the bias towards labels that are compatible with many neighbors' labels is consistent with the probabalistic nature of the rule, it often led to the wrong choices of labels to favor. Normalization of the compatibilities to aleviate this problem is not appropriate since it would alter the relative relationship of disutility between alternative combinations of signal timing plans.

Experiments using the fuzzy product rule were more successful. The bias towards labels which are somewhat compatible with many neighbors' labels, was not present in these tests. Sample runs of the algorithm, which is called MLP, are depicted in Figures 16a and 16b. The general performance (average quality of solutions and average speed of convergence) of MLP is compared to the performance of a few of the single-label IR algorithms in Figure 17. Although convergence of the multi-label IR algorithms cannot be guaranteed because the label confidence update function is too complex for current analysis techniques, the algorithms always converged to a solution, and often converged to a solution quickly.

Optimal solutions were obtained for half of the network configurations. Non-optimal, but reasonably good solutions were obtained for tests on the other, more difficult configurations. Non-optimal solutions were obtained because the confidence values for alternative signal timing plans which were part of an optimal solution, were at some nodes decreased prematurely. Thus, the correct signal timing plans at some nodes were not considered when information supporting the alternatives arrived. We call this problem, which we have come to recognize as a problem inherent with the relaxation technique, the over-enhancement problem.

Modulation, which could slow the changes to confidence values will not prevent over-enhancement because modulation slows the propogation of information. Over-enhancement has the effect of weighing local evidence as to the desirability of signal timing plans more than distant evidence. Thus, although information from some non-neighboring nodes can affect the choice of timing plans for a node, the limited nature of the locality of effect is often apparent, and a global optimal solution can be missed.

In an attempt to understand how much the distributed control contributed to the problem of overehancement, the parallel control structure of MLP was replaced with a serial control structure, resulting in algorithm MLS. For MLS label probability distributions for nodes are updated sequentially, back and forth along a maximal spanning tree. Sequential updating was considered because nodes would be prevented from repeatedly updating their label probability distributions before receiving new information. Information should, therefore, propagate further, and better solutions should be found. Experimental results confirm this



SSP : single-label, serial algorithm
PMLA: single-label, parallel algorithm

MLS : multi-label, serial algorithm
MLP : multi-label, parallel algorithm

Figure 17

conjecture of improved performance with a serial control structure (see Figure 17).

5.2. Discussion

Uncertainties similar to those found in the single-label IR algorithms can be found in the multi-label (relaxation) algorithms. Simultaneous-update uncertainty is present in the parallel relaxation algorithms, but is generally of low magnitude since changes to most label confidence values are limited. Local-view uncertainty can also be found in the relaxation algorithms. A node updates its label probability distribution on the basis of a local view: neighbors' label probability distributions and compatabilities. When its neighbors' label probability distributions incorrectly indicate the global worth of labels, a node may update its label probability distribution incorrectly. As nodes continue to make incorrect updates, over-enhancement becomes a problem.

From a distributed processing viewpoint, a multi-label IR algorithm is more desirable than a single-label IR algorithm. This is because the multi-label IR algorithms require no environmental updating between iterations and should be able to run asynchronously. Although the large label-set size is an inconvenience, the update process is not complicated and can be performed relatively quickly and efficiently. In addition, thresholding of label confidence values can help minimize communication and update time. It also happens that updates become faster as the number of iterations increases because many labels are eliminated.

The success of the relaxation approach in many image processing applications can most likely be attributed to the well distributed nature of initial information, and to weaker interaction among subproblem solutions. For reasons similar to those discussed at the end of Section 4.3, error due to local-view and simultaneous-update uncertainties should be less frequent and of lower magnitude than error due to local-view and simultaneous-update uncertainties found in the NTLC application.

6. Conclusions and Future Directions for Research

A distributed iterative refinement algorithm for network traffic light control produces on the average good but not optimal solutions. In general, convergence to an optimal solution (or to any solution) cannot be guaranteed for many real-world applications using an IR formulation since these formulations are usually in terms of local operators that are discontinuous and not convex. For such applications, the negative effects of these types of operators is significantly reduced by sequentializing the application of the operators in the network. We have found that it is very difficult in a distributed version to reproduce the performance of the sequential versions, in terms of quality of solutions, without significant additional inter-node communication and synchronization.

A single-label IR algorithm can be effective when coupled with good heuristics. However, synchronization requirements and the need for extensive environmental updating before iterations make this approach somewhat inappropriate for a distributed system. Thus, the multi-label IR algorithm appears to be more suited for distributed processing since it does not require inter-node synchronization and environmental updating between iterations. More extensive tests, however, will have to be conducted on a more complex network before the IR algorithms can be effectively judged.

In addition to extensive tests of a larger and more complex network with a more complex platoon flow model and measurements of time, space, and communication costs, studies of reliability or fault tolerance in the face of communication or processing errors should be undertaken. Furthermore, the use of iterative refinement in other problem donains such as air traffic control could be explored. Performance could be significantly better for applications where non-local dependencies are weaker than those of the NTLC application.

One more direction for future research is to examine other approaches to solving the NTLC problem. A promising approach based on the distributed Hearsay II architecture [Lesser and Erman 1979] involves the incremental aggregation of partial solutions. Partial solutions corresponding to sub-network timing plans are formed through this aggregation process. This approach is unlike the IR approach, where we feel there is too much abstraction of information, because processors can see what non-neighboring processors are doing. By keeping explicit track of partial solutions that make up larger partial solutions, non-local interaction among subproblems can be handled correctly and with certainty. The major issues to be faced with this approach include the minimization of communication and redundant or useless processing. We hope to be reporting soon on this

new approach.

Acknowledgments

We wish to thank Dan Corkill, Allen Hanson, and Scott Reed for many fruitful discussions. We also wish to thank Dan Corkill and Scott Reed for their helpful comments on various drafts of this paper.

References

Baudet, G.M. (1976). "Asynchronous Iterative Methods for Multiprocessors," JACM, Vol. 25, p. 226-244.

Baudet, G.M. (1978). "The Design and Analysis of Algorithms for Asynchronous Multiprocessors," Dept. of Computer Science, Carnegie-Mellon University.

Cooper, L. and Steinberg, D. (1970). Introduction to Methods of Optimization, W.B. Saunders Co., Philadelphia.

Dalal, Y.K. (1977). "Broadcast Protocols in Packet Switched Computer Networks," Technical Report No. 128, Digital Systems Lab, Stanford University.

Davis, L.S. and Rosenfeld, A. (1976). "An Application of Relaxation to Spring-Loaded Template Matching," Third Intl. Joint Conference on Pattern Recognition, San Diego.

Davis, L.S. and Rosenfeld, A. (1977). "Hierarchical Relaxation for Waveform Parsing," Computer Science Center, University of Maryland.

DOT/FHA (1976). Traffic Control Systems Handbook, U.S. Department of Transportation / Federal Highway Administration.

Dreyfus, S.E. and Law, A.M. (1977). The Art and Theory of Dynamic Programming, Academic Press, New York.

Erman, L.D. and Lesser, V.R. (1975). "A Multi-Level Organization for Problem Solving Using Many Diverse, Cooperating Sources of Knowledge," Proc. IJCAI-75, pp. 483-490.

Hanson, A.R. and Riseman, E.M. (1978a). "Segmentation of Natural Scenes," Computer Vision Systems, A.R. Hanson and E.M. Riseman (Eds.), Academic Press, New York.

Hanson, A.R. and Riseman, E.M. (1978b). "VISIONS: A Computer System for Interpreting Scenes," Computer Vision Systems, A.R. Hanson and E.M. Riseman (Eds.), Academic Press, New York.

Kilmer, McCulloch, and Blum (1969). A Model of the Vertebrate Central Command System," Int. J. Man-Machine Studies, Vol. 1, pp. 279-309.

Kinney, Kumar, Lin, Ulrich, Petz and Smith (1977). "Distributed Computer Systems for Traffic Control," Prepared by the Department of Electrical Engineering, University of Minnesota, for DOT, Offic of Secretary, Office of University Research.

Lesser, V.R. and Corkill, D.D. (1978). "Cooperative Distributed Problem Solving: A New Approach for Structuring Distributed Systems," COINS Technical Report 78-7, Dept. of Computer and Information Sciences, University of Massachusetts.

Lesser, V.R. and Erman, L.D. (1979). "An Experiment in Distributed Interpretation," Technical report, Computer Science Department, Carnegie-Mellon University.

Lieberman, et al. (1974a). "Variable Cycle Signal Timing Program, Vol. 1: Third Generation Policies for UTCS," KLD Assoc., Inc. for FHA, NTIS PB-241 717.

Lieberman, et al. (1974b). "Variable Cycle Signal Timing Program, Vol. 3: CYRANO: Cycle-Free Responsive Algorithms for Network Optimization: Moderate, Congested, and Light Flow Regimes," KLD Assoc., Inc. for FHA, NTIS PB-241 720.

Lieberman and Woo (1976). "SIGOP II: A New Computer Program for Calculating Optimal Signal Timing Patterns," Transpn. Res. Record, Report 596.

Little, et al (1974). "Variable Cycle Signal Timing Program, Vol. 2: Optimization of Traffic Signals in Networks by Mixed Integer Linear Programming," KLD Assoc., Inc. for FHA, NTIS PB-241 718.

Riseman, E. and Arbib, M.A. (1977). "Computational Techniques in the Visual Segmentation of Static Scenes," Computer Graphics and Image Processing 6, pp. 221-276.

Rosenfeld, Hummel, and Zucker (1976). "Scene Labelling by Relaxation Operations," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-6, pp. 420-433.

Sacerdoti, E.D. (1977). A Structure for Plans and Behavior, American Elsevier, New York.

Spira, P.M. (1977). "Communication Complexity of Distributed Minimal Spanning Tree Algorithms," Proc. Second Berkeley Workshop on Distributed Data Management and Computer Networks.

Tenenbaum, J.M. and Barrow, H.G. (1976). "Experiments in Interpretation Guided Segmentation," SRI Technical Note 123, SRI.

Tate, A. (1977). "Generating Project Networks," Proc. IJCAI-77, pp. 888-891.

Wagner, Barnes, and Gerlough (1971). "Improved Criteria for Traffic Signal Systems in Urban Networks," National Cooperative Highway Research Program Report 124, Highway Research Board, Washington D.C.

Waltz (1975). "Understanding Line Drawings of Scenes with Shadows," The Psychology of Computer Vision, P. Winston (Ed.), McGraw-Hill, New York, pp. 19-91.

Zucker (1978). "Verticle and Horizontal Processes in Low-level Vision," Computer Vision Systems, A.R. Hanson and E.M. Riseman (Eds.), Academic Press, New York.

Zucker, Hummel, and Rosenfeld (1975). "Applications of Relaxation Labeling, 1: Line and Curve Enhancement," TR-419, Computer Science Center, University of Maryland.

Zucker, Leclerc, and Mohammed (1978). "Continuous Relaxation and Local Maxima Selection: Conditions for Equivalence," Computer Vision and Graphics Lab., Dept. of Electrical Engineering, McGill University.

Zucker and Mohammed (1978). "Analysis of Probabalistic Relaxation Labeling Processes," Computer Vision and Graphics Lab., Dept. of Electrical Engineering, McGill University.