交通シミュレータNITTS とそのマルチプロセッサ化について

中村俊一郎、五十嵐智也、宮西洋太郎*、斎藤成一**
（日本工業大学、*公立はこだて未来大学、**三菱電機）

ここ数十年来の懸案である都市及びその近郊における交通渋滞は、人間が社会活動を行う上での大きな阻害要因となっている。数十年来、交通流の解析やシミュレーションに関しても多くの研究がなされてきており、入手可能な交通シミュレータも多い。ただ、今後研究を進めていく上で、様々な角度からきめ細かなシミュレーションを行おうとした時に、やはりソースプログラムを持っているシミュレーショタが必要であろうと考えた。このため昨年より独自の交通シミュレータ NITTS（NIT Traffic Simulator）の開発を進めている。さらに交通シミュレータのアニメーション出力重視という観点から、マルチプロセッサ化の研究開発も進めている。本文では昨年度完成した NITTS1.0 版、及びそのマルチプロセッサ化についての実現方式の概要について述べる。

Development of traffic simulator NITTS and its multiprocessing feature

Shunichiro Nakamura, Tomoya Igarashi, Yohtaro Miyanishi*, Seiich Saito**
(Nippon Institute of Technology, *Future University Hakodate, **Mitsubishi Electric Corp.)

Traffic jams in cities and their suburbs are continuing to be obstructions for human social activity for several decades. Many works analyzing and simulating traffic flow have been developed so far, and there exist several good traffic simulators those we can buy. But from the viewpoint to simulate traffic with various angle and more detail, we thought we should have an own traffic simulator of which we have source codes. So we developing a traffic simulator called NITTS (NIT Traffic Simulator). Also in our point of view that animation outputs are important in traffic simulation, we are developing multiprocessing feature of NITTS. Here we describe about the NITTS 1.0 and its multiprocessing feature.

１．はじめに

都市及びその近郊における交通渋滞は相変わらず深刻な状況が続いており、環境面や人的エネルギー消費の面で、大きなマイナスとなっている。この解決策として、多額な出費が必要な新道敷設や道路拡張の代わりに、近年低廉化の著しい各種情報機器の利用が考えられないだろうかということに我々も注目している。そのため、まずは第一段階として、交通渋滞を解析するための交通シミュレータについて検討した。数十年来、交通流の解析やシミュレーションに関しても多くの研究がなされてきており［１］－［３］、入手可能な交通シミュレータでも素晴らしいものが多数存在している。た

だ、今後研究を進めていく上で、様々な角度からきめ細かなシミュレーションを行おうとした時に、やはりソースプログラムを持っているシミュレーショタが必要であろうと考えた。このため昨年より独自の交通シミュレータNITTS（NIT Traffic Simulator）の開発を進めている［４］。本文では昨年度完成したNITTS1.0 版についてその概要を説明する。次に現在そのマルチプロセッサ化を進めており、その概要について説明する。

２．０　NITTS 1.0 版概要

本章では昨年度完成した NITTS1.0 版の構成と制御方式の概要について述べる。

２.１　データ構造

NITTS の主要データ構造を図１に示す。大きく、道路情報を収容するテーブル群(図２.１上方)と車情報を収容するテーブル群(同下方)に分かれる。

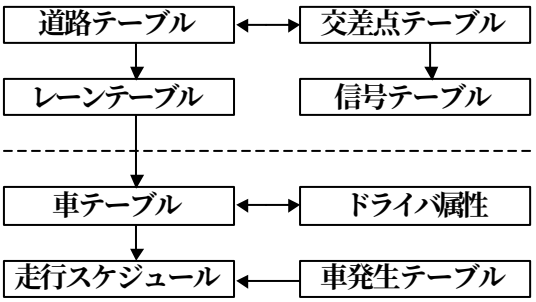


図２.１　NITTS の主要データ

車テーブルはシミュレーション本体が作成するが他のテーブルは事前にユーザが作成する。

２.２　プログラム構造

データ入力プログラムはスタンドアロンで動作するがその他はメインルーチンに統括されシミュレーション実行時に動作する。

シミュレーション本体は、車を動かす動車制御プログラム(２.２)、様々な表示を行う表示系プログラム(２.５)、車を発生させる車両発生プログラム(２.６)、などからなる。

各プログラムを制御しているメインプログラムフローを図２.２に示す。

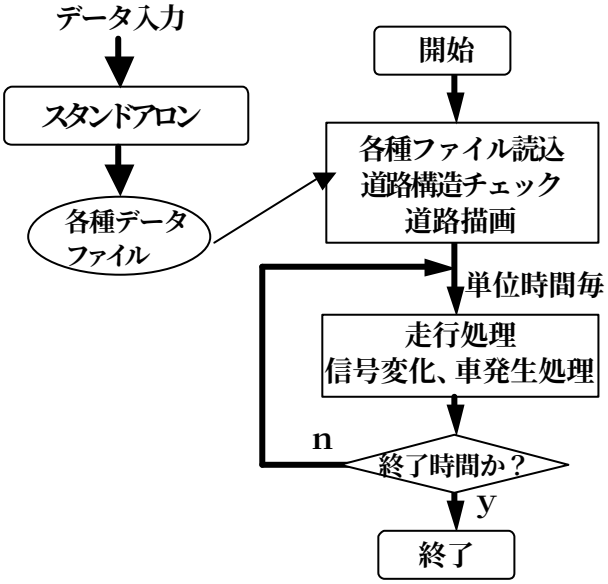


図２.２ メインプログラムフロー

２.３　動車制御プログラム

２.３.１　概要

メインルーチンから１台の車について Call されると、その車の単位時間分の動作をシミュレートする。図２.３に本プログラムの基本フローを示す。

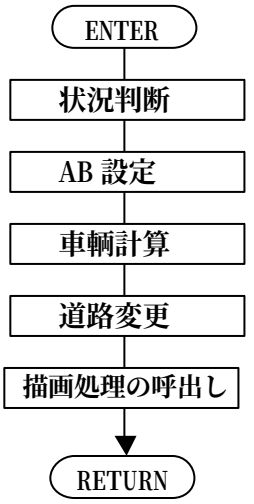


図２.３ 動車基本フロー

状況判断は、その車が置かれている状況を調べ、それに対応した今後の行動を決定し、アクセル/ブレーキ設定（AB 設定）を行うためのものである。図２.４はこの状況の種類を大まかに系統分類したものの一部である。

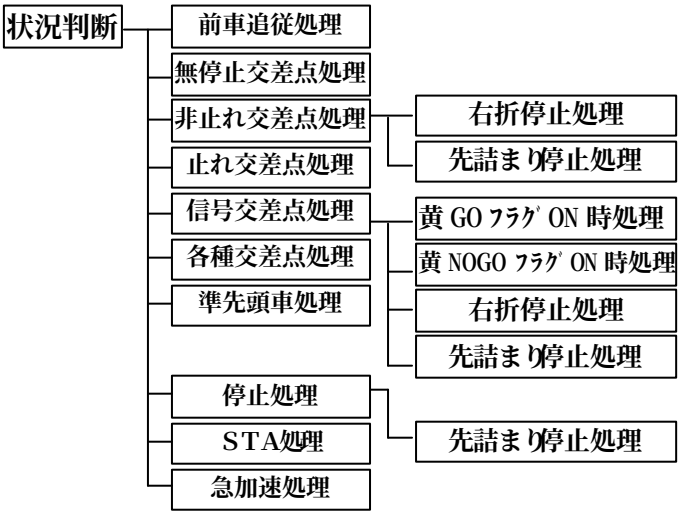


図２.４　状況判断系統図

表２.１　ブレーキ強度表(抜粋)

巡航速度	目標減速値[km/h]				
[km/h]	0	25	20	15	10
100	389	363	371	375	385
70	191	166	174	182	187
50	98	73	82	89	94

単位：[mm/s/0.1秒]

道路変更は自車の変位と道路の長さを比べ、変位が道路長より大きくなっていた場合に行う。その時交差点が終端の場合は、車を消滅させるように値を、終端でなければ次の道路に車を移動させるように値を描画処理に渡す。

２.３.２　車の走行アルゴリズム

車の速度段階は①加速、②定速走行、③減速・停止の３つの構成からなる。前方 100[m] 以内に停止・減速点がなければ①又は②の走行を、停止・減速点がある場合は③の走行を行うことを基本アルゴリズムとした。そこで、停止位置までの間に前車がいない場合は、図５の速度制御アルゴリズムを考案し適用した。

図２.５の実線のグラフを限定速度曲線と名づけ、車はこれに沿って走行する。しかし、この速度制御アルゴリズムでは車 1 台にのみ着目しているので、前車がいた場合、車が衝突してしまう恐れがある。これを回避するため、自車の巡航速度を保ち、かつ安全車間距離を保ちながら前車に追従するアルゴリズムを考え、複数の車でも図２.５のアルゴリズムを応用して速度制御を可能とした。　このために表２.1などを作成し使用した。

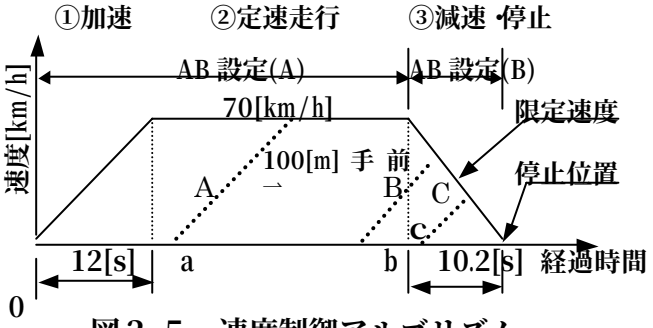


図２.５　速度制御アルゴリズム

２.４　表示系プログラム

２.４.１　概要

シミュレーション結果をグラフィカルに画面表示することは、対人向け出力として最も重要な機能である。そのためシミュレーション結果をリアルタイムに反映する道路描画プログラムと、車両描画プログラムを作成した。

２.４.２　道路描画プログラム

表示系プログラムでは、コンピュータの画面 1 ピクセル（ドット）＝1mと換算している。これにより、Windows の標準的な画面サイズ 1024×768 ピクセルでは、図２.６のように約 1000m×700mの範囲の道路を表示が可能である。よって道路描写プログラムでは以下の機能を作成した。

・ 拡大縮小機能。図２.７は図２.６の拡大時の画面。
・上下３車線までの道路幅に対応。道路幅を青、センターラインを橙色で描画。
・交差点の余分な線と足りない線の処理。

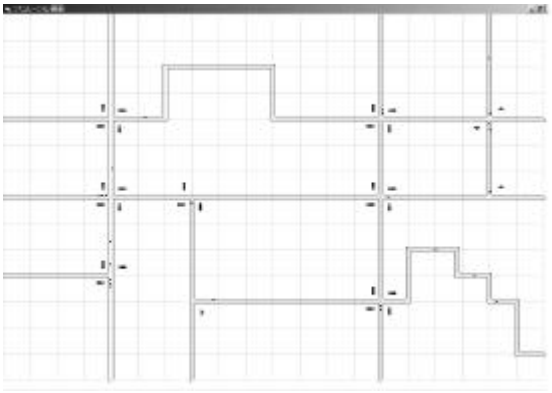


図２.６ シミュレーション画面

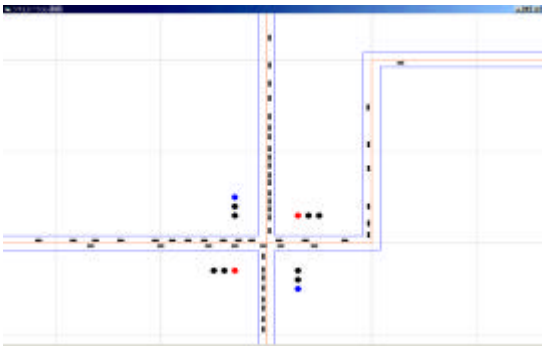


図２.７ 交差点付近５倍での拡大

２.４.３ 車輌描画プログラム

車の表示は乗用車 1 台の大きさを車長 4 メートル、車幅 2 メートルとし、4×2 ピクセルで表示する。また他の車のサイズも考え、全部で 4 種類の表示が可能である。さらに、車一台一台に色を付け、車を区別することを可能としている。

車両描画プログラムは動車系のプログラムから Call されたときに車の座標を計算し、その座標に車を描画する。

２.５ 車輌発生プログラム

２.５.１ 概要

車輌発生処理は①発生時間、②位置、③属性データの３つのパラメータを車輌に持たせ、画面上に車両を発生させるものである。発生した車両は以後、前述の動車処理に従い走行することとなる。

２.５.２ 通常発生処理

メインルーチンから１秒毎に Call され、シミュレーション時間と発生時間が一致した場合に、表示系と動車系に値を渡し車両を発生させる処理である。

２.５３ 初期配置処理

前述の通常発生処理だけでは、シミュレーション開始時には車はおらず、目的の混雑状況になるまで時間がかかる。そのため目標とする状況からシミュレーションを開始するための処理を作成した。

２.６ チェックプログラム

２.６.１ 概要

シミュレータ動作では、多数のデータ入力が必要となるが、必然的に入力ミスが発生し正常なシミュレーションを妨げる。これらのミスを早期に発見し、手戻りを防ぐ目的で道路構造チェックプログラム、標識チェックプログラムを作成した。

２.６.２ 道路構造チェック

交差点から交差点までを一つの道路単位として、長さや関係が正しく、交差点で一致し、平面上に閉じた道路網が構築されているかチェックする。図２.８、図２.９はこのプログラムの実行画面である。図２.８は入力データが正常の場合、図２.９は、道路 a の全長 300m を誤って 200mに入力した場合の出力を示す。

２.６.３ 標識チェックプログラム

信号がなくて交差している道路では必ず一方に“止まれ”標識のあることや、信号交差点において、交差する道路が共に青信号であることがないこと、のチェックを行うプログラムを作成した。

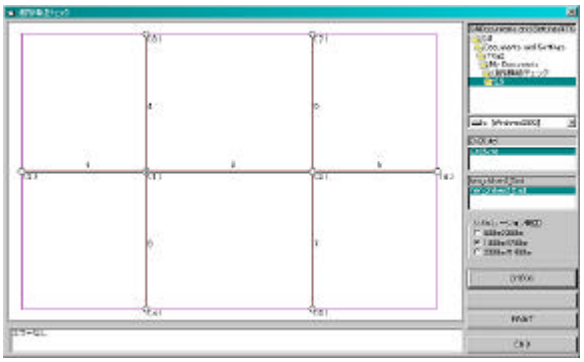


図２.８　入力データが正常の場合

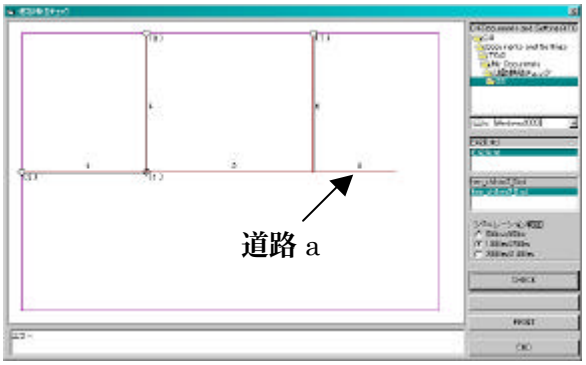


図２.９　道路 a の全長を誤って入力した場合

３．０　NITTS のマルチプロセッサ化について

３．１　マルチプロセッサ化の必要性

交通シミュレータの出力は、大きく分けて以下の２つに分類される。

(1) アニメーション画面

(2) 統計情報（平均交通量、平均通過速度、平均信号待ち回数、走行ログ etc.）

このうち人間にとって直截的で分かりやすいのは前者の方であろう。所与の交通状況において、ある個所の信号制御を変えた場合、その影響（＋の影響と－の影響）が、どのように波及していくかといようなことを、アニメーション画面上で直感的に把握することが出来る。さらに、それにより渋滞状況が隣接する交差点に移ったような場合には、さらにそちらの信号制御を変化させるというように、リアルタイムに調整して試行することが出来る。以上のようにアニメーション画面は、交通シミュレータの出力としてきわめて重要なものと考えることが出来る。

近年のコンピュータの高性能化により、車１台１台をシミュレートするミクロシミュレーションでも、かなりの広範囲な領域をシミュレーションすることが可能になってきた。但しこれは計算上でシミュレーション出来るということであって、アニメーション表示出来ることを意味しない。２章で述べた我々の NITTS1.0 版では、基本画面を 1000m x 700m の領域としているが、人間が見て車 1 台 1 台を判別できるという点では、このあたりの縮尺が限界であることが判明した。この範囲をさらに拡大するための手段としては、例えば 10km x 7km の地図をアニメーション表示させておき（勿論この場合には道のみが表示され、車までは見えない）、その上をズーム画面をずらしながら、あたかも虫眼鏡をずらすようにして見ていくという方法が考えられる。但しこの方法では、ある地点の車の走行状況を見たい時に、その度毎にズーム画面を動かしていって見るという操作が必要であり、わずらわしいと同時に、交通状況を全体的に把握出来るという面ではかなり見劣りすることは否めない。

以上のような問題を解決するために、我々はマルチプロセッサによるマルチ画面表示のアニメーションの手法を考えた。図 3.1 はこの手法の概要を示す図である。即ち、1 台 1 台のパソコンは安くなってきているため、それらを複数用いてマルチプロセッサ化し、各パソコンには 1000m x 700m の範囲のシミュレーションを分担させ、それらのディスプレイを連結して配置することにより、より広域な範囲を同時に画面表示させようとするものである。図 3.1 の例で言えば、都下環状７号線の大原交差点か

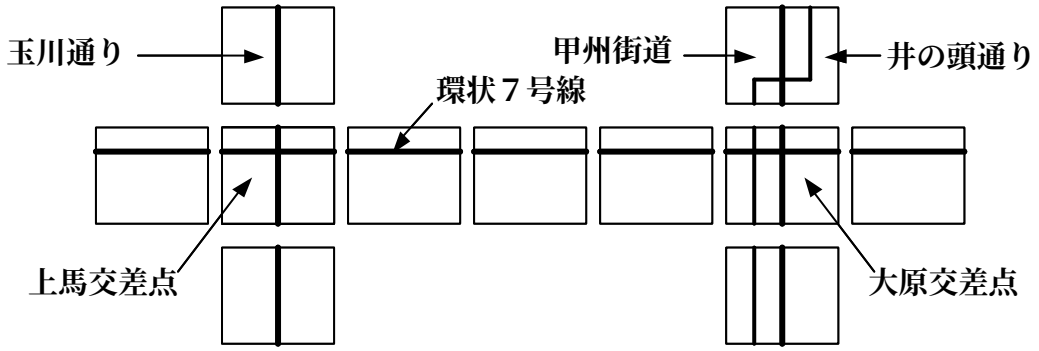


図 3.1 パソコン 11 台のマルチプロセッサ構成によるシミュレーション実施例

ら上馬交差点にかけての、すべての通行車輌を同時実時間表示が可能である。環状７号線の大原交差点を頭とする交通渋滞は、時として上馬交差点あたりまで延びることがあり、そのような渋滞状況の解析を行う場合には、画面がダイレクトに観察出来るため、この方法はきわめて有効である。又この方法ではディスプレイの配置を自由に変えられるため、シミュレーションを行いたい領域の形に合わせてディスプレイを配置できるため、必要なパソコンの台数を最小限に押さえることが出来る利点がある。

以上のような動機に基づき、我々は NITTS のマルチプロセッサ版の研究開発をスタートした。以下にその実現手法等の概要について述べる。

３．２　システム構成

図３.２にハードウェア構成を示す。図示のようにマルチプロセッサを構成するパソコン群は、パソコン相互の通信手段として、100Mbps イーサネットで接続される。通信プロトコルは高速化のために有利な UDP/IP とし、通信ソフトウェアは Winsock コントロールを使用し、これによりプロセッサ間の通信を行うようにした。

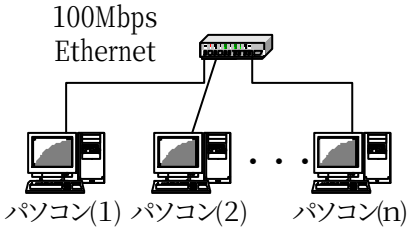


図３.２ ハードウェア構成

３．３　プログラム上の主要な追加要素

以下に上記のマルチプロセッサ・シミュレーションシステムを構成するために考案した、ポイントとなる事項について述べる。

３．３．１　隣接プロセッサ情報テーブル

１つのプロセッサには、上方、右方、下方、左方と最大で４つの隣接するプロセッサが存在し得る。このため各プロセッサは４エントリーからなる隣接プロセッサ情報テーブルを各々保持し、上方、右方、下方、左方にそれぞれ隣接するプロセッサのアドレスを保持する。

３．３．２　隣接道路情報テーブル

隣接するプロセッサをまたがって道路が走っている場合には、それぞれのプロセッサはこの道路が、お互いの相手プロセッサまでつながっていることを意識しなければならない。何故ならこの道路上を自プロセッサ内で走行している車輌が、境界点まで達した時に、相手プロセッサにその旨を伝え、以後当該車輌は自プロセッサ上からは消え、相手プロセッサ上に移って移動していくようにしなければならない。反対に相手のプロセッサから走ってきた車輌が境界点に達した時には、その車輌を自プロセッサ上に移してシミュレーションしなければならない。我々の提案するマルチプロセッサシミュレーションシステムでは、このようにプロセッサ間を車輌が移動する制御は必須な事項で

ある。図 3.3 は１対の隣接するプロセッサと、その間を横断する道路を図示したものである。ここでは３本の道路が横断しているが、プロセッサ間で上記の上方をやりとりするにあたり、この場合３本の道路のうちのどの道路で車輌が移動したかということをお互いに知っている必要がある。このために、それぞれのプロセッサ内で図 3.4 に示すような隣接道路情報テーブルを設けた。NITTS1.0 版からの変更量を少なくするため、道路番号はそれぞれのプロセッサがかってに持っているが、隣接道路情報テーブルはそれらの道路の道路番号を隣接する辺のＸまたはＹ座標の順に並べたものが図 3.4 の隣接道路情報テーブルである。プロセッサ間ではお互いに隣接道路情報テーブルの何行目の道路、という言い方をすることにより、対応する道路をそれぞれ結びつけることが出来る。

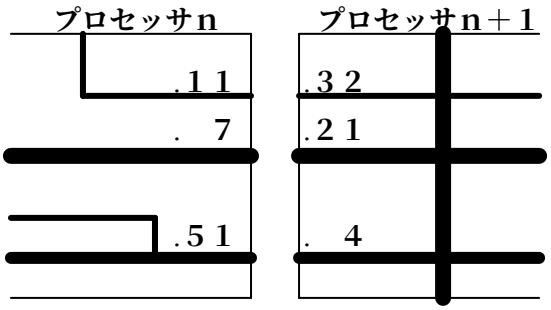


図３.３　隣接なプロセッサ

プロセッサｎ		プロセッサｎ＋１
道路番号１１	⟷	道路番号３２
道路番号　７	⟷	道路番号２１
道路番号５１	⟷	道路番号　４

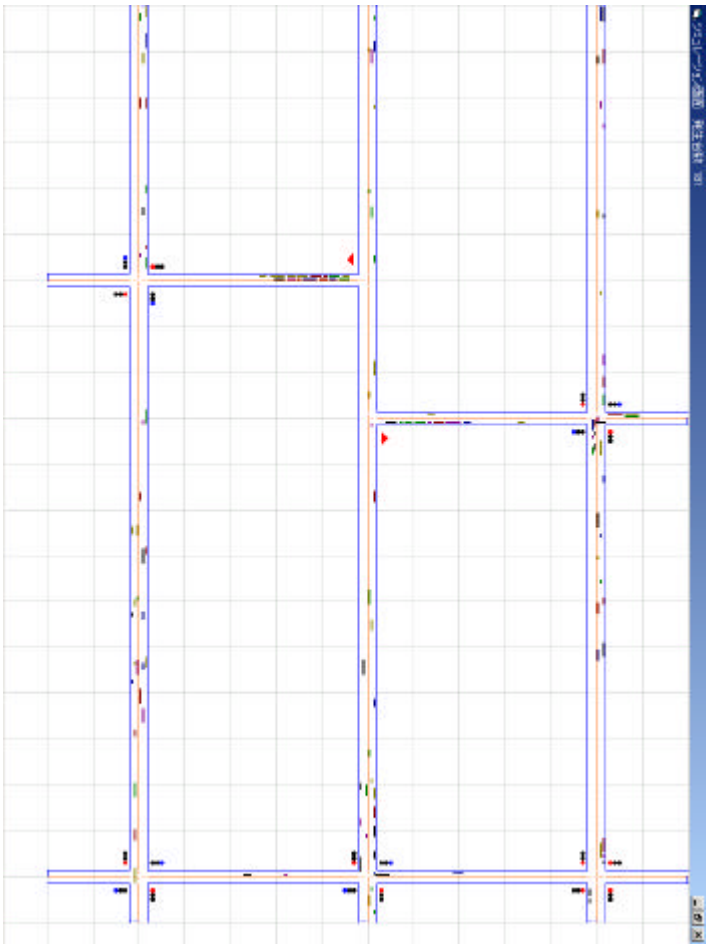
図３.４　隣接道路情報テーブル

３．３．３ 進入速度指定テーブル

１つのプロセッサの中であれば、プログラムはシミュレーション上のあらゆるデータに自由にアクセスすることが可能であるが、２つのプロセッサに別れた場合それは中々大変になる。勿論相手プロセッサ内の情報が必要になった都度、通信手段を用いてそれを得るようにすることは可能であるが通信量が膨大になってしまう。例えば、車輌が連結道路の接続点に向かって走っている時に、その先の相手側プロセッサの道路上の端の近傍に遅い車輌が走っていれば、その車輌との車間距離を考慮しながら減速して走行しなければならない。反対に何も走っていなければフリー走行の速度で進入すればよい。このようなことをまともにやって、自プロセッサ内で車を動かす毎に相手プロセッサ内の情報をいろいろ調べるようにしたら、通信量は膨大になる。この問題を解決するために、我々は進入速度指定テーブルという近似的手法を考案した。これは、車輌に進入される側のプロセッサから、その道路（レーン）の接続点に進入する時には、どういう速度で進入せよと、指定を行うものである。進入速度指定テーブルでは、図 3.4 と異なり１エントリーが道路単位でなくレーン（車線）単位になっている。例えば片側２車線道路であれば、左側車線は渋滞中であるが右側車線は空いていてフリー走行が可能というような場合も有り得るからである。プロセッサは隣接する道路のすべてのレーンの進入速度を設定し、相手側プロセッサに知らせる。シミュレーションは 0.1 秒毎のタイムクォンタムでなされるが、この進入速度指定テーブルの転送は１秒毎に行われるようにした。

３．３．４　試行結果

以上のような基本方式によりマルチプロセッサ版の開発を行っている。まだ試行段階であるがその実行例を図 3.5 に示す。同図に示される２つのプロセッサ間を接続する道路上で、車輌がプロセッサ間を移行して、同一プロセッサ内走行と同様に、ごく自然に走行することが観察されている。



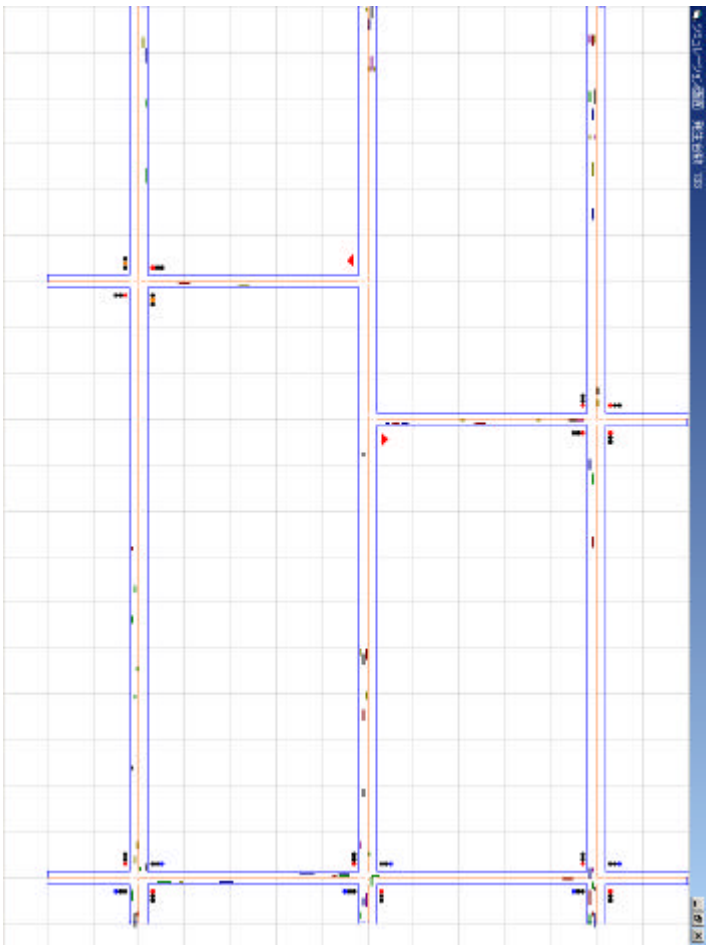


図３．５　２プロセッサ構成時のマルチプロセッサ版シミュレータの実行アニメーション画面

４．０　おわりに

交通シミュレータ NITTS の 1.0 版の概要を延べ、それをマルチプロセッサ化する提案を行った。マルチプロセッサ化のための基本方式を検討し、試作を行い良好な結果を得た。今後はこのマルチプロセッサ版の完成を目指して努力していく所存である。

参考文献

［1］　桑原雅夫：広域ネットワーク交通流シミュレーション, 自動車技術, Vol.52, No.1, pp28-34（1998）

［2］　猪飼國夫，本多中二：ファジィモデルに基づく市街地での渋滞予測用微視的シミュレータ，日本ファジィ学会誌, Vol.11, No.2, pp215-221（1999）

［3］　斎藤威：交通渋滞予測のための道路交通現象の再現, 電気学会誌, Vol. 117, No.9, pp600-603（1997）

［4］　五十嵐智也, 中村俊一郎，斎藤成一，上田尚純，宮西洋太郎，吉田幸二：道路交通シミュレータ NITTS の開発, 日本シミュレーション学会，第 20 回シミュレーションテクノロジーコンファレンス，10-4, pp295-298（2001）