GPU Based Micro-simulation and Optimization for Artificial Transportation Systems

Dr. Zhen SHEN
Associate Professor
zhen.shen@ia.ac.cn

State Key Laboratory of Management and Control for Complex Systems
Beijing Engineering Research Center of Intelligent Systems and Technology
Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
Outline

- Overview
- Micro-simulation
- Optimization
- Conclusions
Urban Traffic Signal Control System (UTSCS) is the important part of the ITS. Without changing the traffic facilities, UTSCS can alleviate traffic congestion by using information, communication, and control technologies.

- Beijing: up to 4.89 million vehicles by April 2011
- Beijing population: reached approximately 20 million by the end of 2010
- A single commuting trip typically takes **52 minutes**.
- A big challenge for the Intelligent Transportation Systems (ITS) research

Evaluation and optimization of ITS

Based on actual traffic system

Impact traffic safety, inefficient, and difficult to implement

Based on the simulation system

Difficult to establish the system model
## Traffic simulation

<table>
<thead>
<tr>
<th>Macro and Meso</th>
<th>Micro</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lighthill–Whitham–Richards (LWR)</td>
<td>Cellular Automata (CA)</td>
</tr>
<tr>
<td>Lattice Boltzmann methods (LBM)</td>
<td>Multi-Agent System (MAS)</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

### Traffic Optimization

- **General Optimization Algorithms**
  - Linear programming, simplex method …

### Traffic Simulation

- **Macroscopic**
- **Mesoscopic**
- **Microscopic**

### Heavier Computational Burden

- Macro- and meso-models lack the capability of describing the details of system and are no longer popular when the computer technologies become developed.

- Micro-simulation systems:
  - MATSim, VISSIM, TRANSIMS, TransWorld, MITSIM, MITSIMU, CORSIM, SHIVA and UTOBAHN
Parallel Systems Theory

- Real transportation system
- Artificial transportation system
- Transportation operator and administrator training system (OTSt)
- Configuration evaluation and verification system (DynaCAS)
- Traffic control and management system (aDAPTS)

ACP method

- **Artificial Systems**: Establish the microscopic model of the human, vehicles and roads in the transport system.

- **Computational Experiments**: Based on the operating results of the artificial transport system, evaluate and optimize the performance of the control configuration.

- **Parallel Execution**: The obtained optimized configuration will be applied to the real system, then adjust the artificial system and the optimization configuration according to the implementation results.
Artificial transportation system

- For the microscopic model, a so called “Artificial Transportation Systems (ATS)” is built in a bottom-up way, with drivers, vehicles, roads, traffic lights being modeled as autonomous, collaborative and reactive agents.

- Build the separate models of all travelers using the agent technology.

- Generate the travel path and the behaviors of car following and lane changing
- Generate the agents with different ages, genders, occupations and so on
- Generate travel activities of agents
- Generate artificial population
- Data and supporting module
- Microscopic traffic simulation

- Data display and analysis
- Display and analyze the results of simulation
- Generate travel plans according to agents’ attributes
- Management, storage and application of traffic historical data and measured data.

The usage of the agent technology
The modeling

It is necessary to find a more efficient parallel method better using computing resources

**Decomposition of functions:**
Different computing resources can be applied to different modules in the different subsystems or the same subsystems.

**Decomposition of space:**
- The large-scale road network in the traffic subsystem is divided into sub-networks.
- Then the tasks of computing the vehicles’ state in the subnet are assigned to different computing resources.

<table>
<thead>
<tr>
<th>Synchronization</th>
<th>Microscopic traffic simulation is synchronized by the same simulation clock. Each subroutine is required to be synchronized in each simulation step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing load balancing</td>
<td>When every simulation subroutine keeps synchronous, the speed of the simulation of entire road network is limited by the node having the heaviest computing resource</td>
</tr>
<tr>
<td>Communication</td>
<td>When the number of the agents that travel cross the sub-network is large, the efficiency of the program is limited by the delay of the LAN communication</td>
</tr>
</tbody>
</table>
A GPU based multi-agent model

Why GPU?
Computing density of GPU is much larger than the current mainstream CPU
— GPU is the parallel computing resources that have the largest computing density
Communication delay of GPU is much shorter than the LAN’s
— Reduce the performance loss of communication through increasing computing density

Development of Computing Resources

- Single-core CPU
- Multi-core CPU, Computer cluster
- CPU+GPU
  - NVIDIA CUDA

GPU
- A specialized circuit
- Originally designed to offload graphics tasks from CPU
- Appears on video card or mother board
- Program with OpenGL and Cg

General-Purpose GPU (GPGPU)
- Compute Unified Device Architecture (CUDA) developed by NVIDIA
- Program with C, C++, Fortran
Kernel: a function that executes on the GPU is typically called a “kernel”.

Hierarchical structure:
- **Hardware**: GPU, SM, SP
- **Software**: Grid, Block, Thread

CPU: control the process, sequential work
GPU: parallel computing
Main idea and motivation:

Our idea is to take use of the most suitable hardware and apply the most suitable algorithms, and then the most up-to-date solutions can be obtained.

We believe: GPU (or other parallel computing tools) will enable iterative, parallel algorithms to prevail.

Contribution:

- Use GPU to parallelize the micro-simulation and optimization of a traffic system
- Show: this integration of GPU with Multi-Agent Systems (MAS) and parallel iterative algorithms can help solve real problems more practically

Parallel micro-simulation: MAS model

Parallel optimization: Genetic Algorithms (GA), Ordinal Optimization (OO), NSGA-II and Vector OO (VOO) for traffic signal timing optimization problems
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
</tr>
<tr>
<td>Micro-simulation</td>
</tr>
<tr>
<td>Optimization</td>
</tr>
<tr>
<td>Conclusions</td>
</tr>
</tbody>
</table>
Micro-simulation

Traffic supply:
1. The road network is composed by intersections, overpasses and city highway transport infrastructure.
2. Given road network topologies before the simulation, and no change during the simulation.

Traffic demand:
1. Agents perform travel activities based on their own travel plans.
2. Search for the optimal path between two adjacent location.

The number of iterations is set to 50 and 10% of the agents re-plan their routes based on the average link travel times of the last iteration.

Hierarchical relationship:
The road network consists of more than one intersection. Each intersection connects with many roads leading to it. Each road contains more than one lane. Vehicle agents travel on every lane. Therefore, the basic unit of the transport facilities for carrying vehicles is lane in the road network.
GM-type car following model:
In the earliest stimulus-reaction model, the stimulus is the velocity of a vehicle agent and the changes of the relative velocity and the relative distance between two adjacent vehicles, while the reaction factor is the acceleration of the agent.

\[ a_n(t + \tau_n) = \alpha \frac{v_n(t + \tau_n)^\beta}{[x_{n-1}(t) - x_n(t)][v_{n-1}(t) - v_n(t)]} \]

- \( a_n(t) \), \( v_n(t) \), \( x_n(t) \): acceleration, speed and position
- \( \tau_n \): the driver’s reaction time
- \( \alpha \), \( \beta \), \( \gamma \): constant parameters

Lane change model:

The intention to change the lane
Select the target lane, check the feasibility
Change the lane

GIPPS model calculate the acceptable retarded velocity:
\[ b_n = [2 - (D - x_n(t)) / (10V_n)] \cdot b_{LC} \cdot \theta \]

- \( b_n \): the acceptable deceleration
- \( D \): the predictive position after changing the lane
- \( x_n(t) \): the position of the agent at the time \( t \)
- \( V_n \): the desired velocity of the agent
- \( b_{LC} \): mean value of all agents’ acceptable deceleration
- \( \theta \): the agents’ attribute parameters
The vehicle agents plan the routes, make the car following and lane changing decisions.

- Initialize the iteration time, and generate routes of travel for all agents
- Initialize the simulation clock
- Refresh the states of traffic lights
- The simulation clock increases by one step
- Refresh travel route
- Iteration count increases by one
- Output simulation results

GPU

- Refresh the position, speed and acceleration of all vehicle agents based on car following model
- Perform lane change for the vehicle agents that have needs based on lane change model
Microscopic traffic simulation parallelization

The allocation of computing resources:
1. Grid: one-dimensional block
2. Each block processes one road
3. One thread in a block is responsible for calculating the acceleration, velocity and position of one vehicle agent in the road network.

The allocation of storage resource:
1. Put the data of the vehicle agent into the global memory of GPU, then copied to shared memory of corresponding block, respectively
2. Put the traffic light data into constant memory of GPU.
### Experiment

1. **Lattice road network**

   - **CPU:** Dual Intel SandyBridge six-core, 2.0G Hz E5-2620
   - **GPU:** Two Tesla C2075
   - **Memory:** 16 * 8GB DDR3 ECC
   - **Iteration number:** 50

   In each iteration we run the simulation for 10,800 seconds. Departure time of the agents obey the normal distribution \( N(3000 \text{ s}, 1000) \)

5,000 vehicles all-together

<table>
<thead>
<tr>
<th>Road network</th>
<th><strong>CPU Only</strong> (Average Time/s, Standard Deviation)</th>
<th><strong>CPU+GPU</strong> (Average Time/s, Standard Deviation)</th>
<th><strong>Speedup</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>5 × 5</td>
<td>100.65, 10.21</td>
<td>54.86, 2.33</td>
<td>1.83</td>
</tr>
<tr>
<td>10 × 10</td>
<td>321.53, 13.94</td>
<td>58.44, 0.57</td>
<td>5.50</td>
</tr>
<tr>
<td>20 × 20</td>
<td>1195.48, 20.21</td>
<td>59.16, 0.96</td>
<td>20.21</td>
</tr>
<tr>
<td>40 × 40</td>
<td>4504.85, 35.79</td>
<td>59.76, 0.78</td>
<td>75.38</td>
</tr>
</tbody>
</table>
Lattice road network

Comparison between GPU and CPU simulation in 50th iteration

Comparison between 1st and 50th iteration of GPU simulation
2. Zhongguancun road network (18 intersections)

<table>
<thead>
<tr>
<th></th>
<th>Only CPU</th>
<th>CPU+GPU</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average Time/s</strong></td>
<td>89.29</td>
<td>56.80</td>
<td>1.57</td>
</tr>
<tr>
<td><strong>Standard Deviation</strong></td>
<td>7.34</td>
<td>1.93</td>
<td>N/A</td>
</tr>
</tbody>
</table>
3. Beijing Second Ring road network (185 intersections)

<table>
<thead>
<tr>
<th></th>
<th>Only CPU</th>
<th>CPU+GPU</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Time/s</td>
<td>593.92</td>
<td>57.13</td>
<td>10.40</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>9.68</td>
<td>2.24</td>
<td>N/A</td>
</tr>
</tbody>
</table>
**Optimization**

**Parameters in traffic control**

*Signal phase:* Combination of traffic lights’ lamp color, every combination is called a phase of the signal.

1. **Phase sequence:** the order of the different phases
2. **Cycle:** the time for a intersection to traverse all phases at a pre-set order
3. **Green ratio:** the ratio of effective green time and cycle
4. **Phase difference:** the difference of the green starting time of the same phase, between two intersections with the same phase sequence

In the control sub-region, the parameters of all traffic signals are set to form a coordinated control scheme.

*Optimization:* optimize the parameters of all traffic signals to achieve the better performance of the road network.
### Performance indicators

In order to identify the congestion degree scientifically, evaluate and compare the pros and cons of the different control configurations. Then improve and optimize them by introducing reasonable performance indicators.

<table>
<thead>
<tr>
<th><strong>Throughput of road network</strong></th>
<th>under specific road conditions and traffic control configuration, the maximum number of vehicles that leave the road network in unit time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average delay</strong></td>
<td>The difference between the time vehicles travel through the intersection in the real case and the time vehicles travel the same distance with the desired velocity</td>
</tr>
<tr>
<td><strong>Average stop time</strong></td>
<td>the number of parking when vehicles are traveling through the intersection</td>
</tr>
</tbody>
</table>

- For single vehicle, the smaller its delay is, the less its stop time will be.
- For single intersection, the total stop time do not become less when the total delay become smaller.
- There is not only a certain relevance between average delay and average stop time in the intersection. They can be used as two performance indicators that are relatively independent to evaluate the pros and cons of the traffic control system.
The multi-objective optimization aim to find out the solution that meet the multiple optimization objectives at the same time. The performance improvement of a target may cause performance degradation of the other one or several other ones.

The solution in the search space is $\theta$, and the objectives of the multi-objective optimization are $J(\theta)$.

1. **Domination:** Assuming the target number of a multi-subjective optimization problem of minimizing is $m$. If there is $J_i(\theta_1) \leq J_i(\theta_2), i = 1, 2, ..., m$ for all targets and at least one sign of inequality that is strict, say $\theta_1$ dominates $\theta_2$.

2. **Incomparable:** If $\exists i, j, i \neq j$, s.t. $J_i(\theta_1) < J_i(\theta_2)$ and $J_j(\theta_1) > J_j(\theta_2)$, say $\theta_1$ and $\theta_2$ are incomparable. That is unable to judge which one is better.

There are special solutions in the solution space of the multi-objective optimization problem.

1. They are incomparable to each other.
2. And there is no other solution to dominate these solutions.

A collection of these solutions is **Pareto frontier** that is the solution of the multi-objective optimization problem.
We investigate two traffic signal coordination problems:

- One is the single objective problem maximizing the number of vehicles leaving the road network in a given time.
  1. GPU-adaptive Parallel Genetic Algorithm (GA)
  2. A Famous method for the large scale system optimization called “Ordinal Optimization (OO)”

- The other is a two-objective problem minimizing the delay time and stop times of vehicles.
  1. Non-dominated Sorting GA II (NSGA-II)
  2. Vector Ordinal Optimization (VOO)

The optimization can be viewed simply as a smart way of organizing multiple runs of simulations with different configurations.
**Case 1:** Lattice road network

**Case 2:** The road network of Zhongguancun

---

Phase 1 | Phase 2 | Phase 3 | Phase 4
---|---|---|---
[Diagram of a lattice road network]

Intersection 1 | Intersection 2 | Intersection 3 | Intersection 4

Phase 1 Phase 2 Phase 3 Phase 4

Intersection

Phase 1Phase 2Phase 3

---

26
**Single-objective:**

Max the number of vehicles leaving a road network in given time period.

### Single-objective optimization model

1. Assuming a control sub-area contains $N$ intersections, the traffic signal at each intersection has $M$ control phase, and all traffic signals have the same communal cycle $c$.
2. If we choose No. 1 intersection to be the reference then the phase difference between every signal light and the reference intersection light can be expressed as

   \[
   \bar{\psi} = [\psi_1, \psi_2, \ldots, \psi_N]^T, \psi_1 = 0
   \]

3. The duration of green of each phase of all signal lights:

   \[
   \bar{\theta}_m = [\theta_{1m}, \theta_{2m}, \ldots, \theta_{Nm}]^T, m = 1, 2, \ldots, M
   \]

   \[
   \theta_{nm}, n = 1, 2, \ldots, N —— the green time of the $m$-the phase in the $n$-the intersection
   \]

<table>
<thead>
<tr>
<th>$\max f(c, \bar{\psi}, \bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_M)$</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s.t. c_{\min} \leq c \leq c_{\max}$</td>
<td>$f(c, \bar{\psi}, \bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_M)$</td>
</tr>
<tr>
<td>$0 \leq \bar{\psi} \leq ce_N,$</td>
<td>$\bar{\theta}_{\min,m} (m = 1, 2, \ldots, M)$</td>
</tr>
<tr>
<td>$\bar{\theta}<em>m \geq \theta</em>{\min,m}e_N, m=1,2,\ldots,M$</td>
<td>$c_{\min}, c_{\max}$</td>
</tr>
<tr>
<td>$e_N$</td>
<td></td>
</tr>
</tbody>
</table>

---

27
Single-objective: Genetic Algorithm

GPU-based parallel genetic algorithm

Encoding of the control configuration, generate initial population $P$ of the coordinated control configuration

Evaluate all configurations in $P$ using the GPU parallel technology

For every configuration $I_i$ in $P$, select another configuration random. Generate $I_i'$, and put it in the position $i$ in $W$

Perform mutation operation for individuals in $W$

Evaluate all configurations in $W$ using the GPU parallel technology

Comparing $I_i$ and $I_i'$, then add the better one to the department $i$ in $P$

We evaluate the average value of the throughput which is obtained by the micro-simulation repeatedly.

Crossover and mutation operations

| Generation 1 | 1 | 0 | 0 | 1 | 1 | 0 | ... | 1 | 1 | 0 |
|-------------|---|---|---|---|---|---|     |   | 1 | 1 | 0 |
| Generation 2| 1 | 1 | 1 | 0 | 0 | 0 | ... | 0 | 1 | 0 |
| Offspring   | 1 | 0 | 0 | 1 | 1 | 0 | ... | 0 | 1 | 0 |
| Offspring   | 1 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 1 | 0 |
### Single-objective: Genetic Algorithm

The parameters of every traffic light need \((M+2)\) 8-bit binary code. The total length of \(N\) signal encoding is \(N \times (M+2)\)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Decoding method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cycle</strong></td>
<td>[ c = c_{\text{min}} + \phi_1 \frac{c_{\text{max}} - c_{\text{min}}}{255} ]</td>
</tr>
<tr>
<td><strong>Phase difference</strong></td>
<td>[ \psi_n = \phi_2 \frac{c}{255} ]</td>
</tr>
</tbody>
</table>
| **Green time in Phase \(M\)**    | \[ p_{mn} = p_{\text{min}} + \phi_3 \frac{p_{\text{max}} - p_{\text{min}}}{255}, m = 1, 2, ..., M \]  
  \[ \theta_{mn} = \theta_{\text{min},m} + \frac{p_{mn}}{M} \left( c - \sum_{i=1}^{M} \theta_{\text{min},i} \right), m = 1, 2, ..., M \] |

\(\phi_m = \sum_{k=0}^{7} 2^k b_{mk}, m = 1, 2, ..., M + 2,\)

\(b_{mk}\) —the value of the \(k\)-th bit in the \(m\)-th 8-bit binary code.
Overall implementation

CPU CODES
- Generate first generation of traffic signal timing configurations
- Launch Kernel1
- Launch Kernel2
- Generate a new generation of traffic signal timing configurations
- Exit?
- Output the best so far configuration

GPU CODES
- Evaluate every configuration with traffic flow simulator
- Selection, Crossover, and Mutation

Data structures and computing resources allocation

A grid for $C$ configurations
- $T$ independent replications for each traffic lights configuration
- Each row of blocks: a replication of configuration
- A block: vehicles in the same lane
- A thread: a vehicle

CPU
- Configuration 1
- ... Configuration $C$
- Intersection 1
- Intersection 2
- ... Intersection $N$
- Road 1
- Road 2
- ... Road $R$
- Lane 1
- Lane 2
- ... Lane $L$
- Vehicle 1
- Vehicle 2
- ... Vehicle $V$

Global Memory

Constant Memory

GPU

Traffic Lights Data

Vehicle Data

Grid
- Lane1 Lane2 Lane3
- ... Lane1 Lane2 Lane3
- ... Lane1 Lane2 Lane3

$T$ replications for Configuration $I$

$T$ replications for Configuration $C$
Case 1: Genetic Algorithm under Single-objective

**Settings**

- \( N = 4 \), number of intersections
- 6 lanes with 3 at either side for a road
- Vehicle chooses new lanes with equal prob., i.e., \( 1/3 \)
- Distance between two neighboring intersections: \( 256 \times 4 = 1024m \)
- Length of a vehicle: \( 4m \)
- Vehicle arrival rate: \( \lambda = 0.2 \) vehicle/s for each entry lane
- Initial speed: \( N(60\text{km/h}, 10) \)
- Desired speed: \( 80\text{km/h} \)
- Parameters: \( \alpha = \beta = \gamma = 1 \)
- \( M = 4 \), number of phases
- \( c_{\text{max}} = 240s, c_{\text{min}} = 60s \): for cycle time
- \( p_{\text{min}} = 0, p_{\text{max}} = 100 \)
- \( \Theta_{\text{min}}, m = 6s \ (m = 1, 2, \ldots, M) \)
- Time step: 1s, simulate for 3600s
- \( C = 500 \), number of chromosomes
- \( T = 10 \), replications
- \( P_c = 0.99 \), crossover
- \( P_m = 0.05 \), mutation
- 1,000 generations

**Speedup results**

Time consumption of one iteration in GA

<table>
<thead>
<tr>
<th></th>
<th>CPU only</th>
<th>CPU + GPU</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Time/s</td>
<td>1197.08</td>
<td>20.47</td>
<td><strong>58.48</strong></td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>10.79</td>
<td>0.89</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Road network throughput: a typical run**
Case 2: Genetic Algorithm for single-objective

- **Four parameters of traffic signal**
  1. The longest cycle is $c_{\text{max}} = 240s$.
  2. The shortest cycle is $c_{\text{min}} = 60s$.
  3. The maximum and minimum value of phase difference are $\theta_{\text{min},m} = 6s$, ($m = 1, 2, 3, 4$).
  4. The shortest green time of every phase is $\theta_{\text{min},m} = 6s$, ($m = 1, 2, 3, 4$).

- The number of iteration of GA is set to 1,000. The population of configurations is 500. The time of repeated evaluations for each coordinated control configuration is 10. The probability of crossover and mutation in GA are 0.95 and 0.05 respectively.

The computation time for single iteration have been calculated based on 30 independent replicatioins.

<table>
<thead>
<tr>
<th>Traffic Volume</th>
<th>Iteration Time</th>
<th>CPU based serial configuration</th>
<th>CPU+GPU based parallel configuration</th>
<th>Acceleration rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>19500</td>
<td>0</td>
<td>2445.66</td>
<td>103.09</td>
<td>23.72</td>
</tr>
<tr>
<td>20000</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20500</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21000</td>
<td>300</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21500</td>
<td>400</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22000</td>
<td>500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The overall time-consuming of the GPU optimization program still reached about 103,090s!
**Ordinal comparison:** It is easier to determine which of the given two designs is better than to obtain the accurate performance difference.

**Goal softening:** In the case of the search space being too large and very difficult to evaluate, we aim at good enough designs (top $n\% \approx 5\%$) rather than the best for sure design.

**Step 1:** Sample the entire searching space to obtain the sampling set $N$. Then evaluate every individual in $N$ to determine the general structure of the solution space.

**Step 2:** Obtain the selecting set $S$ as the object for accurate evaluation in $N$, namely after evaluating $S$, we can obtain the solutions which are in the top $n\%$ in the entire solution space with high probability.

$$P_A^* = P\left\{|G_\Theta \cap S| \geq k\right\} \geq 0.95$$

$k$: alignment level
Single-objective: Ordinal Optimization

Algorithm implementation steps:

1. Using the same encoding method with GA, generate 1,000 binary codes randomly which constitute the sample set $N$.

2. Use the decoding method to obtain all parameters in all configurations of $N$.

3. Give the parameters to the agents of the traffic lights, then run the artificial system to obtain the specific values of the traffic volume under different configurations. (Rough evaluation model: evaluate once for each configuration)

4. Order the 1,000 performance values obtained from the rough evaluation to obtain the Ordered Performance Curve (OPC), then determine their types.

(1) **Flat**: There are many good performance designs in the solution space
(2) **U-Shaped**: There are more good and bad designs in the solution than medium design
(3) **Neutral**: The number of designs of every style is similar to each other
(4) **Bell**: More medium designs than good and bad
(5) **Steep**: There are many bad designs
Single-objective: Ordinal Optimization

5. According to the judgment of the OPC type, determine the number of the control configurations in the assemblage \( S \) through the look-up table using the formula:

\[
s = Z(k, g) = e^{z_1} k^{z_2} g^{z_3} + Z_4
\]

<table>
<thead>
<tr>
<th></th>
<th>Flat</th>
<th>U-Shaped</th>
<th>Neutral</th>
<th>Bell</th>
<th>Steep</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z_1 )</td>
<td>8.5200</td>
<td>8.2232</td>
<td>8.4832</td>
<td>8.8697</td>
<td>8.2995</td>
</tr>
<tr>
<td>( Z_2 )</td>
<td>0.8944</td>
<td>0.9426</td>
<td>1.0207</td>
<td>1.1489</td>
<td>1.3777</td>
</tr>
<tr>
<td>( Z_3 )</td>
<td>-1.2286</td>
<td>-1.2677</td>
<td>-1.3761</td>
<td>-1.4734</td>
<td>-1.4986</td>
</tr>
<tr>
<td>( Z_4 )</td>
<td>5.00</td>
<td>6.00</td>
<td>6.00</td>
<td>7.00</td>
<td>8.00</td>
</tr>
</tbody>
</table>

6. Evaluate all individuals in the \( S \) using the accurate model, ensuring that the probability of 0.95 for obtaining \( k \) solutions that are in the top \( n\% \) of the search space, wherein \( g=1,000 \times n\% \). (Accurate evaluation model: ten evaluations for each configuration)

7. For large-scale control sub-area of road network, the searching space is too large. In order to ensure the “uniformity” of the samples, the operations from step 1 to 6 can be repeated until obtain a performance to meet the requirement of the coordinated control configuration.
Overall implementation

CPU CODES

Obtain the sample space \( N \)

Launch Kernel 1

According to the judged type of OPC, determine the number of the schemes

Launch Kernel 2

Evaluate the schemes in \( N \) infrequently (Rough evaluation model)

GPU CODES

Evaluate the schemes in \( S \) frequently (Accurate evaluation model)

Evaluation Model

Optimization Model

Data structures and computing resources allocation

CPU

Configuration 1

Intersection 1

Road 1

Lane 1

Turning Lane 1

Vehicle 1

Traffic Lights Data

Vehicle Data

Configuration C

Intersection 1

Road k1

Lane i1

Turning Lane i1

Vehicle j1

Grid

Lane 1

Lane 2

Lane 2\( ^*T*1/k/l_{1} \)

T replications for Configuration 1

T replications for Configuration 2

T replications for Configuration

Shared Memory

GPU

Constant Memory
Case 1: Ordinal Optimization under Single-objective

Set the sampling set $N$ contain 1,000 coordinated control programs. The time of evaluation for every configuration is 1 in the rough evaluation model, while it is 10 in the accurate evaluation model.

\[ k = 1 \quad g = 1000 \times 1\% = 10 \]

\[ Z_1 = 8.8697 \quad Z_2 = 1.1489 \quad Z_3 = -1.4734 \quad Z_4 = 7.0 \]

\[ s = e^{z_1} k^{z_2} g^{z_3} + Z_4 \]

The size of the selected set $S$ is $s=45$, i.e. evaluate the top 45 coordinated control configurations accurately in $N$.

Relative to the genetic algorithm, the coordinated control configuration obtained by OO will is about 3.5% worse.

GA: 20,470 s vs. OO: 70 s
Case 2: Ordinal Optimization for Single-objective

Set the sampling set $N$ containing 1,000 coordinated control programs. The time of evaluation for every configuration is 1 in the rough evaluation model, while it is 10 in the accurate evaluation model.

$$k = 1 \quad g = 1000 \times 1\% = 10$$

$$Z_1 = 8.8697 \quad Z_2 = 1.1489 \quad Z_3 = -1.4734 \quad Z_4 = 7.0$$

$$s = e^{z_1 k z_2 g z_3} + Z_4 = 45$$

Evaluate the top 45 coordinated control configuration accurately in $N$.

Relative to the genetic algorithm, the coordinated control configuration obtained by OO is about 3.3% worse.

GA: 103,090 s vs. OO: 284 s
Multi-objective:
A two-objective problem minimizing the delay time and stop times of vehicles.

Multi-objective optimization model

Minimizing the delay time and stop times are two typical and contradictory optimization objective

The cycle $c$ of the signal affects the optimization objectives:
1. If the signal cycle is shorter, the average delay is shorter while the average stop times increases.
2. If the signal cycle is longer, the average delay is longer while the average stop times becomes less.

$$\begin{align*}
\min \quad & F=[f_1(c, \bar{\psi}, \bar{\vartheta}_1, \bar{\vartheta}_2, \ldots, \bar{\vartheta}_M), f_2(c, \bar{\psi}, \bar{\vartheta}_1, \bar{\vartheta}_2, \ldots, \bar{\vartheta}_M)] \\
\text{s.t.} \quad & c_{\min} \leq c \leq c_{\max} \\
& 0 \leq \bar{\psi} \leq ce_N, \\
& \bar{\vartheta}_m \geq \theta_{\min,m} e_N, m = 1, 2, \ldots, M,
\end{align*}$$

Minimizing the delay time and stop times of vehicles are the two objectives.
## Multi-objective: Non-dominated Sorting GA II

### Algorithm implementation steps:

1. Using the same encoding method to the GA, generate first generation random, then put it into $P_t$.

2. Use the decoding method to initialize control parameters of all configurations, then run micro-simulation to obtain the specific average values of the delay time and stop times.

3. Calculate the Pareto frontier solution of the individual $p$ in $P_t$.

4. In order to maintain the diversity of individuals, calculate the virtual fitness $p_{distance}$ of the individual $p$ in $P_t$.

5. Selection operation: Select two individuals $p$ and $q$ in $P_t$. If $p_{rank} < q_{rank}$, or $p_{rank} = q_{rank}$ and $p_{distance} > q_{distance}$, choose $p$ as the object of the crossover and mutation operations until the scale of the population reaches $N$.

6. Obtain the temporary offspring population $Q_t$ by the crossover and mutation operations.

7. Elite strategy: Constitute $R_t$ by combining $P_t$ and $Q_t$. Repeat step 3 to 5 for $2N$ individuals, then select first $N$ individuals to constitute the offspring population $P_{t+1}$.

8. Repeat step 3 to 7 until meeting the exit criteria.
Overall implementation

CPU CODES
- Generate first generation randomly
- Launch Kernel
  - Calculate the level of non-inferior solution and the virtual fitness of all individuals
- Generate a new generation

Evaluation Model
- Exit?
- Y
- Output the best so far configuration

Optimization Model

GPU CODES
- Evaluate all configurations, then calculate the average values of delay and stop times with traffic flow simulator

Data structures and computing resources allocation

CPU
- Traffic Lights Data
- Global Memory

GPU
- Constant Memory
- Grid
  - Lane 1
  - Lane 2
  - Lane k1
  - Lane i1
  - Lane j1

Vehicle Data
- Vehicle 1
- Vehicle 2
- Vehicle j1

T replications for Configuration 1
- T replications for Configuration 2
- T replications for Configuration m
Case 1: Non-dominated Sorting GA II for Multi-objective

The iterative time is 500, the size of the population in the control configuration is 200, the repeated time of the evaluation in every coordinated control configuration is 10, the probability of crossover and mutation is 0.95 and 0.05 respectively.

The data about the computation time of single iteration are obtained based on 30 independent runs.

The overall time-consuming of the GPU optimization program still reached about 51,780s!
**Case 2:** Non-dominated Sorting GA II for Multi-objective

The iterative time is 500, the size of the population in the control configuration is 200, the repeat time of the evaluation in every coordinated control configuration is 10, the probability of crossover and mutation is 0.95 and 0.05 respectively in GA.

The data about the computation time of single iteration are based on 30 independent runs.

<table>
<thead>
<tr>
<th></th>
<th>CPU based serial configuration</th>
<th>CPU+GPU based parallel configuration</th>
<th>Acceleration rate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average time/s</strong></td>
<td>3886.67</td>
<td>140.62</td>
<td><strong>27.64</strong></td>
</tr>
<tr>
<td><strong>Standard deviation</strong></td>
<td>27.35</td>
<td>2.97</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The overall time-consuming of the GPU optimization program still reached about 140,620s!
Multi-objective: Vector Ordinal Optimization

VOO indexes the layers of Pareto solutions and take out the top $s$ layers as selected set.

Evaluate all the solutions in the sampling set $N$ and obtain the selected set $S$. $x$ is the layer number, and $F(x)$ is the number of solutions in the top $x$ layers. Then we can get the Ordered Performance Curve in the vector case (VOPC).

(1) Flat type: The Pareto frontier contains only a few solutions. This is the difficult case.
(2) Intermediate type: The difficulty level is between flat and steep type.
(3) Steep type: The Pareto frontier contains many good solutions. This is the easiest.
## Multi-objective: Vector Ordinal Optimization

**Algorithm implementation steps:**

1. Specify the number of individuals $M$, the layer count of the satisfying solutions $g$, the number of real satisfying solution $k$ in the desired results, and alignment probability (usually 0.95).

2. Use the same encoding method, and generate $M=1,000$ binary code randomly which constitute the sample set $N$.

3. Use decoding method to obtain parameters in all configurations of $N$.

4. Give the control parameters to the agents of the traffic lights, then run the artificial system to obtain the specific values of the traffic volume under different configurations. *(Rough evaluation model: evaluate once for each configuration)*

5. Evaluate and sort the designs in $N$, by the rough model to obtain the VOPC, then determine their types.

6. According to $g$, $k$, $M$ and the count of layers $\widehat{s_0}$ obtained by stratifying $N$, adjust the values of $g$, $k$.

\[
g' = \max \left\{ 1, \frac{100}{\widehat{s_0}} \times g \right\} \quad k' = \max \left\{ 1, \frac{10000}{M} \times k \right\}
\]

\[\cdot\] ——Floor operator
7. Use the formula \( s' = Z(k, g) = e^{z_1}k^{z_2}g^{z_3} + Z_4 \) to determine the initial value of the size \( s' \) in the selected assemblage \( S \).

<table>
<thead>
<tr>
<th>( Z )</th>
<th>Flat type</th>
<th>Intermediate type</th>
<th>Steep type</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Z_1 )</td>
<td>5.2099</td>
<td>0.9379</td>
<td>0.3885</td>
</tr>
<tr>
<td>( Z_2 )</td>
<td>0.9220</td>
<td>0.8445</td>
<td>0.8536</td>
</tr>
<tr>
<td>( Z_3 )</td>
<td>-1.9542</td>
<td>-0.8890</td>
<td>-0.8847</td>
</tr>
<tr>
<td>( Z_4 )</td>
<td>1.9662</td>
<td>0.5946</td>
<td>0.5414</td>
</tr>
</tbody>
</table>

8. According to \( s' \), obtain the size of the selected assemblage \( S \).

\[
\hat{s} = \left\lfloor \frac{\hat{s}_0 \times s'}{100} \right\rfloor
\]

\( \left\lfloor \cdot \right\rfloor \) —— Ceiling operator

9. Evaluate the solutions in the first \( s \) layers accurately, to ensure the probability of 0.95 for obtaining \( k \) satisfying solutions at least in the first \( g \) layers. (Accurate evaluation model: ten evaluations for each configuration)
Overall implementation

CPU CODES
- Obtain the sample collection $N$ and specified $T, g, k$
- Launch Kernel 1
- According to the results of evaluation, stratify all the configurations in N to obtain VOPC
- Based on the empirical formula, adjust values of $g, k$
- Launch Kernel 2
- Output the satisfying solutions of the multi-objective optimization problem

GPU CODES
- Evaluate the schemes in $N$ infrequently (Rough evaluation model)
- Evaluate the schemes in $S$ frequently (Accurate evaluation model)
- Evaluation Model
- Optimization Model

Data structures and computing resources allocation

CPU
- Traffic Lights Data
- Global Memory
- Constant Memory

GPU
- Vehicle Data
- Grid
- Lane 1
- Lane 2
- Lane $k_l$

$T$ replications for Configuration 1

$T$ replications for Configuration 2

$T$ replications for Configuration $m$
Case 1: Vector Ordinal Optimization for multi-objective

Set the sampling space $N$ to contain 1,000 designs. The number of replication of evaluation for every configuration is 1 in the rough evaluation model, while it is 10 in the accurate evaluation model.

All solutions are divided into $s_0 = 58$

The layer count of the satisfying solution $g = 1$

The number of solutions in the real Pareto frontier $k = 1$

$g' = 1, k' = 10$

$Z'' = Z(k, g) = e^{z_1}k'^{z_2}g'^{z_3} + Z_4 = 17.72$

$\hat{s}_0 \times \frac{s'}{100} = 11$

GPU based GA:
About 51,780 s

GPU based OO:
About 307s
Case 2: Vector Ordinal Optimization for multi-objective

Set the sampling set $N$ contain 1000 coordinated control program. The time of evaluation for every configuration is 1 in the rough evaluation model. While it is 10 in the accurate evaluation model.

All solutions are divided into $s_0 = 44$

The layer count of the satisfying solution $g = 1$

The number of solutions in the real Pareto frontier $k = 1$

$$g' = 2 \quad k' = 10$$

$$s' = Z(k, g) = e^{z_1 k' z_2} g' z_3 + Z_4 = 10.24$$

$$s = \left[ \frac{s_0}{100} \times s' \right] = 5$$

GPU based GA: About 140,620 s

GPU based OO: About 613s
Conclusions

Future research

- Improve the micro-simulation model: rotary island, highway
- Solve large-scale road network problems with Message Passing Interface (MPI)
- Use GA and OO with some local search algorithms to improve the performances
- Different criteria should be considered: mean travel time of the vehicles, mean length of the queues, and multi-objective optimization.
Thank You!

Dr. Zhen SHEN  
Associate Professor  
Tel: +86–10–82614437  
Fax: +86–10–82615087  
Email: zhen.shen@ia.ac.cn  
Institute of Automation, Chinese Academy of Sciences