A GPU-Based Parallel Genetic Algorithm for Generating Daily Activity Plans

Kai Wang and Zhen Shen, *Member, IEEE*

***Abstract*—As computing technologies develop, there is a trend in traffic simulation research in which the focus is moving from macro- and meso-simulation to micro-simulation since micro-simulation can provide more detailed quantitative results. Moreover, the success of the Artificial societies-Computational experiments-Parallel execution (ACP) approach indicates that integrating other metropolitan systems such as logistic, infrastructure, legal and regulatory, and weather and environmental systems to build an Artificial Transportation System (ATS) can be helpful in solving Intelligent Transportation Systems (ITS) problems. However, the computational burden is very heavy as there are many agents interacting in parallel in the ATS. Therefore, a parallel computing tool is desirable. We think that we can employ a Graphics Processing Unit (GPU), which has been applied in many areas. In this paper, we use a GPU-adapted Parallel Genetic Algorithm (PGA) to solve the problem of generating daily activity plans for individual and household agents in the ATS, which is important as the activity plans determine the traffic demand in the ATS. Previous research has shown that GA is effective but that the computational burden is heavy. We extend the work to GPU and test our method on an NVIDIA Tesla C2050 GPU for two scenarios of generating plans for 1000 individual agents and 1000 three-person household agents. Speedup factors of 23 and 32 are obtained compared with implementations on a mainstream CPU.**

***Index Terms*—Artificial societies–Computational experiments–Parallel execution (ACP), artificial transportation system (ATS), compute unified device architecture (CUDA), daily activity plan, genetic algorithm (GA), graphics processing unit (GPU), microsimulation.**

Manuscript received February 27, 2012; revised June 8, 2012; accepted June 13, 2012. Date of publication July 19, 2012; date of current version August 28, 2012. This work was supported in part by the National Natural Science Foundation of China under Grant 60921061, Grant 70890084, Grant 90920305, Grant 90924302, Grant 60904057, Grant 60974095, and Grant 31170670 and in part by the Chinese Academy of Sciences under Grant 2F09N05, Grant 2F09N06, Grant 2F10E08, Grant 2F12N02, Grant 2F11D03, Grant 2F11E08, and Grant 2F10E10. The Associate Editor for this paper was F.-Y. Wang.

K. Wang is with the Center for Military Computational Experiments and Parallel Systems Technology, College of Mechatronics Engineering and Automation, National University of Defense Technology, Changsha 410073, Hunan Province, China, and also with the 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.

Z. Shen (Corresponding author) is with the 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, and also with Dongguan Research Institute of CASIA, Cloud Computing Center, Chinese Academy of Sciences, Songshan Lake, Dongguan 523808, Guangdong Province, China (e-mail: zhen.shen@ia.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2012.2205147

I. Introduction

TRAFFIC simulation is important in intelligent transportation system (ITS) research [1]. As computing technologies develop, the focus is changing from macro- and mesosimulation to microsimulation. Among the microsimulation models, the multiagent system is one of the most important in traffic system modeling, analyzing, and forecasting [2]–[4]. In this model, vehicles, bicycles, and pedestrians are modeled as agents that are autonomous, collaborative, and reactive. The agents can communicate, compete, and cooperate with each other, and macroscopic traffic phenomena can be observed and analyzed [5]. One main advantage of the model is that it can simulate the impacts of traffic control and management methods of the ITS on the agents' daily travels in a quantitative way [6]. Moreover, other metropolitan systems, such as logistic, infrastructure, regulatory, and weather systems, can also be considered in the microsimulation, which is the idea behind the artificial transportation system (ATS) [1], [5]. Based on the ATS, the ACP approach was proposed to model, analyze, and control the transportation systems [1]. It consists of three steps: 1) modeling and representing using the ATS; 2) analyzing and evaluating by *C*omputational experiments; 3) controlling and managing through *P*arallel execution of real and artificial systems. There have been many successful applications of ACP in ITS [1], [4], [5], [7]–[11].

Generating daily activity plans for the agents is an elementary problem in the ATS as the plans determine the traffic demands. In the real world, people tend to quickly and easily make reasonable activity plans, which is not easy for the computers. If there are ten activities, the number of choices is a permutation of 10, which is 3 628 800, and the computer has to judge which of them are reasonable. Recently, genetic algorithm (GA) has been shown effective in solving this problem, and previous research has tested it for an individual [12] and a household agent [13], [14], respectively. However, the computation burden is heavy when multiple agents' plans are simultaneously generated. In this paper, we employ a graphics processing unit (GPU)-adapted parallel genetic algorithm (PGA) to perform computational experiments to generate activity plans in a parallel way.

II. Literature Review

A. Generating Daily Activity Plans and GA

When people make their daily activity plans, usually, they first list all activities possibly to be done and then select and order the activities according to location, starting time,

duration, as well as route and travel mode of the activities and the influences of the subsystems in building an ATS. The selections and assignments are all based on their knowledge on whether a plan is "good" or not. People perform their activity plans by travels, which form the traffic flow.

However, it is not an easy task for computers to generate the plans as we need to make the computers understand which plans are "good." Usually, there are two main prerequisites for the computers: the first one is to prepare activity lists of all possible activities that the agents tend to do. The second one is to design proper utility functions to evaluate the activity plans. Based on the two prerequisites, different algorithms can be employed to select activities from the list and make them scheduled. One of the most mature and popular methods is the discrete choice model (DCM), which is a statistical procedure that models choices made by people among a finite set of alternatives by estimating the probability that a person chooses a particular alternative. However, it has to enumerate all possible plans, and even unreasonable plans may be selected [6], [12]. Therefore, other algorithms such as Monte Carlo chains [15], generic rules [16], mental maps [17], and GAs [12], [18] are employed. Among the new methods, GA has been shown to be effective in generating daily activity plans for an individual or a household agent, and it is flexible enough to easily change the utility function according to different preferences of the agents and different conditions of the subsystems in the ATS [12]–[14], [18]. The GA method has already been used in MATSim, which is an open-source software development project developing an agent-based traffic microsimulation module [12]–[14], [18], [19].

However, there is a great challenge in computing that we should not only evaluate a large number of plans for thousands of generations for a single agent but also have to complete the task of generating daily activity plans for all agents involved in the ATS. It is almost impossible for conventional desktop computers to complete the computing task within an acceptable time [12]. To make matters worse, important urban events such as hosting the Olympic Games, urban population migration, traffic policy adjustments, and changes in traffic control and management programs always result in different travel demands [10]. Activity plans are necessary to be regenerated in different computational experiments. Therefore, it is desirable to use powerful computing tools to accelerate the GA method.

B. GPU and CUDA

As described in our previous papers [5] and [20], GPU is a specialized circuit that has many cores working together. The cores are called streaming processors, and several cores (8 or 32, typically) are organized into a streaming multiprocessor (SM). Each SM has its own memory called shared memory, in which all threads in it can simultaneously access, and all SMs share the global memory, constant memory, and texture memory of the GPU.

In software, a typical GPU program consists of two parts: one part is the CPU codes that control the process of the whole program and does the sequential work, and the other is the GPU part that does the parallel work. With compute unified device architecture (CUDA), the programmers can use C-style codes to access the computing resources of the GPU, and the programming on GPU does not have much difference from using application programming interfaces. A function that executes on the GPU is typically called a "kernel" [20]. When a kernel is launched, multiple threads on the GPU organized by two levels are activated. The top level is called "grid," and the other is called "block." One grid can consist of at most 65 535 $\times$ 65 535 blocks, and each block can consist of up to 1024 threads in the Fermi architecture.

III. Formulation and Implementation

A. Problem Formulation and Overall Implementation

For generating a daily activity plan, we need to decide whether an activity is selected from the activity list and decide the orders of the selected activities. For an activity, a person may arrive early, leave early, arrive late, or leave late. The key point is to give proper utility functions and maximize the total utility. We consider two kinds of agents separately: One is the individual agent, and the other is the household agent. For a household agent, some activity may be performed by members of the family together, and others may be done by specific members of the family. To demonstrate the utility function, we design a virtual city that consists of several kinds of activity facilities with different opening times, and every facility is distributed in several different locations. We follow the model given in [12], [13], and [18] to set up the utility functions.

First, we describe how to formulate the problem of generating a daily activity plan for an individual agent. The utility of the activity plan can be described as follows:

$$U_{\text{total}} = \sum_{i=1}^{k} U_{\text{dur},i} + U_{\text{wait},i} + U_{\text{late.ar},i} + U_{\text{early.dp},i} + U_{\text{short.Dur},i} + U_{\text{travel},i} \quad (1)$$

where U_{total} is the total utility of an individual agent; k is the total number of activities in the activity plan; $U_{\text{dur},i}$ is the utility of performing the ith activity in the plan; $U_{\text{wait},i}$ is the (dis)utility of waiting for the facility of the ith activity to open; $U_{\text{late.ar},i}$ and $U_{\text{early.dp},i}$ are the (dis)utilities of arriving late and leaving early from the facility, respectively; $U_{\text{short.dur},i}$ is the (dis)utility of executing the activity for a duration that is too short; and $U_{\text{travel},i}$ is the (dis)utility of the individuals traveling from the location of the (i-1)th activity to the location of the ith activity. We define

$$U_{\text{dur},i} = \beta_{\text{dur}} \cdot t_{W,i} \cdot \ln\left(\frac{t_{\text{dur},i}}{t_0}\right) \quad (2)$$

$$t_0 = t_{W,i} \cdot e^{-c_i \big/ (t_{W,i} \cdot p_i \cdot \beta_{\text{dur}})} \quad (3)$$

where $t_{W,i}$ is the typical duration of the ith activity, $t_{\text{dur}.i}$ is the actual duration of the activity, p_i is the priority of the activity (which is a number from the set $\{1, 2, 3\}$, with "1" being the highest priority), β_{dur} is the marginal utility of an activity at its typical duration (which can be verified by calculating $dU_{\text{dur}.i}/dt_{\text{dur}.i}$), c_i is the coefficient of the activity duration, and $U_{\text{dur}.i}$ is logarithmic to the ratio of its actual duration $t_{\text{dur}.i}$ and the breakeven duration t_0. $U_{\text{dur}.i}$ increases when $t_{\text{dur}.i}$ is

longer [12]. Equations (2) and (3) provide a heuristic way for measuring the utility

$$U_{\text{wait},i}=\begin{cases}\beta_{\text{wait}}(t_{\text{loc.open},i}-t_{\text{start},i}), & \text{if } t_{\text{loc.open},i}>t_{\text{start},i}\\ 0, & \text{otherwise}\end{cases} \tag{4}$$

where $t_{\text{start},i}$ is the starting time of the ith activity, $t_{\text{loc.open},i}$ is the open time of the facility of the activity, and β_{wait} is the coefficient of waiting.

$$U_{\text{arr\_late},i} = \begin{cases}\beta_{\text{late.ar}}(t_{\text{start},i}-t_{\text{latest.ar},i}), & \text{if } t_{\text{start},i}>t_{\text{latest.ar},i}\\ 0, & \text{otherwise}\end{cases} \tag{5}$$

where $t_{\text{latest.ar},i}$ is the latest starting time of the ith activity, and $\beta_{\text{late.ar}}$ is the coefficient of arriving late at the facility.

$$U_{\text{early.dp},i} = \begin{cases}\beta_{\text{early.dp}}(t_{\text{earliest.dp},i}-t_{\text{end},i}), & \text{if } t_{\text{earliest.dp},i}>t_{\text{end},i}\\ 0, & \text{otherwise}\end{cases} \tag{6}$$

where $t_{\text{earliest.dp},i}$ is the earliest ending time of the ith activity, $t_{\text{end},i}$ is the actual ending time and $\beta_{\text{early.dp}}$ is the coefficient of leaving early from the facility.

$$U_{\text{short.dur},i} = \begin{cases}\beta_{\text{short.dur}}(t_{\text{shortest.dur},i}-t_{\text{dur}.i}), & \text{if } t_{\text{shortest.dur},i}>t_{\text{dur}.i}\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

where $t_{\text{shortest.dur},i}$ is the shortest duration of the ith activity, and $\beta_{\text{short.dur}}$ is the coefficient of short duration of the activity.

$$U_{\text{trav},i}=\beta_{\text{trav}}\cdot t_{\text{trav},i} \tag{8}$$

where $t_{\text{trav},i}$ is the time consumption from the facility of the (i-1)th activity to the facility of the ith one, and β_{trav} is the coefficient of traveling.

$t_{W,i}$, p_i, $t_{\text{loc.open},i}$, $t_{\text{latest.ar},i}$, $t_{\text{earliest.dp},i}$, $t_{\text{shortest.dur},i}$, and all the coefficients can be assigned with different values according to different agents' preferences and different conditions of the subsystems of the ATS. $t_{\text{trav},i}$ can be calculated from the length of the route and the travel speed of the agents.

Second, we take family relationship into consideration and give the formulation for a household agent following [14]. It is assumed in the model that household members may do some specific activities together, and some activities can be allocated to different household members to represent division of work. To allow GA to generate plans for all household members, it is necessary to extend the utility function to account for additional utility derived from joint participation in certain activities. This way, we take into account how household members interact and synchronize with each other. The utility function of activity generation for household agents can be described as follows:

$$F=\sum_{m=0}^{M}U_{\text{total},m} \tag{9}$$

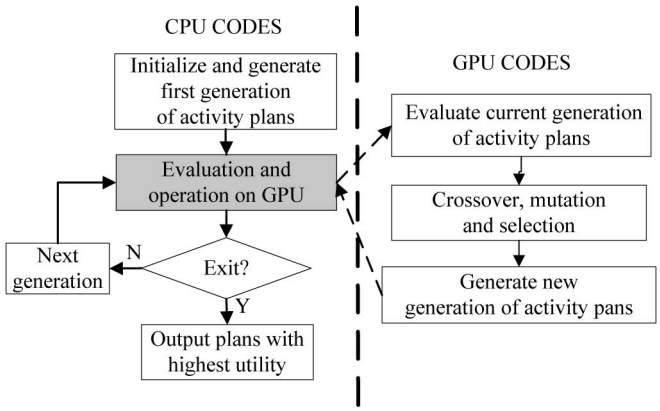

Fig. 1. Overall implementation with the GPU.

where F is the utility of a household agent, M is the number of the members of the household, and $U_{\text{total},m}$ is the utility of the mth member of the household. If the ith activity is to be performed by all the members of the households, $U_{\text{dur}.i}$ of the activity of one member can be modified as follows:

$$U_{\text{dur}.i}=U_{\text{dur}.i}\left(1+\beta_{\text{joint},i}\cdot\frac{t_{\text{dur.share}}}{t_{\text{dur}.i}}\right) \tag{10}$$

where $t_{\text{dur.share}}$ is the overlapping parts of all members' $t_{\text{dur}.i}$. $\beta_{\text{joint},i}$ is the coefficient of the joint activity. The more a person is synchronized with other household members in the activity, the closer $t_{\text{dur.share}}/t_{\text{dur}.i}$ is to 1, and the higher the additional utility is [14].

In a transportation system, an individual or a household is relatively independent from others [12], [14]. Therefore, the activity plans of multiple agents can be generated in parallel. Moreover, for one individual or household agent, we tend to use PGA to maximize (1) or (9) that a random population of the agent's activity plans is generated at the beginning of the algorithm. One plan is independent from the others, and the population of plans can be evaluated in parallel. Our implementation contains a two-level parallelization: One is the parallelization of different agents, and the other is parallelization of one agent's different plans. The overall implementation is shown in Fig. 1.

B. PGA

The activity plans of one agent can be processed by multiple threads of the GPU in parallel with operations of selection, crossover, and mutation. Recently, there has been a growing interest in implementing the PGA with GPU. We employ the PGA, which has been proven to be effective in solving the quadratic assignment problem [21] and the traffic signal timing optimization problem [20]. In this PGA, two pools P and W of the same size are used to store current and newly generated offspring individuals. The algorithm flow is described as follows.

Step 1) Generate an initial population of individuals of P.
Step 2) Evaluate the individuals in P.
Step 3) For each individual I_i in P, randomly select another individual $I_j(i\neq j)$ in P. Apply crossover to I_i and

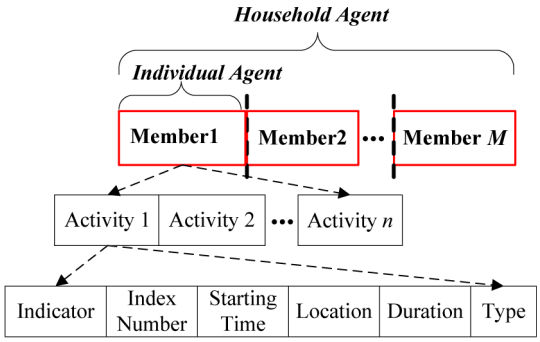

Fig. 2. Genetic representation.

I_j with probability P_c. If the crossover happens, put the offspring I_i' into W; otherwise, copy I_i into W.

Step 4) For each individual I_i' in W, apply the mutation with probability P_m.

Step 5) Evaluate individuals in W.

Step 6) For each individual I_i in P and its corresponding offspring or copy I_i' in W, compare the fitness values. If I_i' has a higher fitness, replace I_i in P with I_i'.

Step 7) If the termination criteria are met, terminate the algorithm. Otherwise, go to Step 3.

Steps 2–6 all have the same operations on individuals. This is why GA can be parallelized.

1) Genetic Representation: A daily activity plan is formed by an ordered series of activities selected from the list of all possible activities. We use an array to encode an agent's activity list as a chromosome of the PGA. For each activity, there are an activity indicator, index number in the activity list, location, duration, and type of the activity. The activity indicator is a binary variable indicating whether the activity is included in a plan or not. The location indicates the coordinate of the activity facility in a city map. If the starting time of the first activity is given, the orders, durations of the activities, and travel time determine the starting time and ending time of all activities in the plan. For a household agent, the type of an activity is used to indicate whether the activity is jointly performed with other household members or individually. There are usually several members in a household, and we merge arrays of the plans of all household members to form a longer array that is shown in Fig. 2.

2) Initial Settings: We generate an initial population of plans for the agents by randomly selecting activities from the activity list and setting their orders. For one plan, the indicators of the selected activities are assigned with "true." We further assign the starting time of the first activity with a random number that obeys $U[0, 24, hour]$ and assign the durations of all the activities of the plan with random numbers that obey $N(t_W, \sigma_{t_W}^2)$, with t_W representing the typical duration time. Locations are randomly selected from the location list. For the household agents, the types are also assigned with different values, which indicate whether they are jointly or individually performed.

3) Crossover and Mutation: The operations of crossover, mutation, and selection are only performed on a single member's array. Different members of a household exchange information via evaluating $U_{\mathrm{dur}.i}$, which is the utility of performing

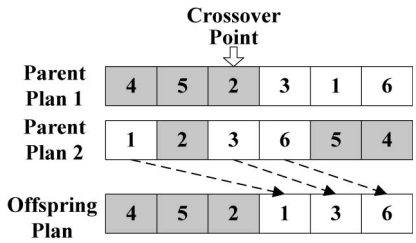

Fig. 3. Crossover operator.

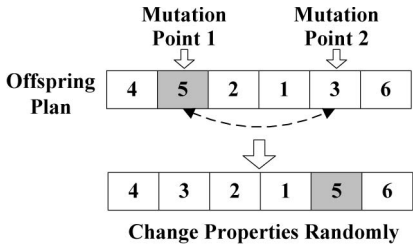

Fig. 4. Mutation operator.

joint activities. We assume that there is no duplication of activities in each plan. When we take the one-point crossover operator on a pair of parent plans, the same activity may appear twice in the offspring. To solve this problem, we take activities before the crossover point of the first parent plan and directly pass them to the offspring plan. Then, we fill up the remaining elements of the offspring plan with the remaining activities in the sequence that are in the second parent plan. The procedure is shown in Fig. 3, in which the numbers are the index numbers of the activities on the activity list.

Furthermore, we perform the mutation operation on the offspring plan by exchanging two activities that are randomly selected. Then, we randomly select activities in the plan and reassign their properties such as the starting time and duration in the same way of generating the initial population of activity plans. The procedure is shown in Fig. 4.

C. Computing Resources Allocation

For both the individual and household agents, the data of the activity plans are organized from top to bottom in two levels, i.e., agents and their activity plans. The agents' initial population of plans is generated on the CPU side and allocated to a grid with one dimension of blocks at the GPU side. As shown in Fig. 5, one block handles one individual agent or one member of a household agent, and one thread handles its one activity plan. Before the kernel function on the GPU is launched, the data of activity plans and parameters of the utility function are copied from the CPU to the GPU. The data of plans are copied to the global memory of the GPU and then copied to the shared memory of the SMs as the access to the shared memory is faster. The parameters of the utility function are copied to the constant memory as we have assumed that all agents share the same utility function to evaluate their plans and that the constant memory can be read by all the threads in the grid with lower memory access latency than the global memory.

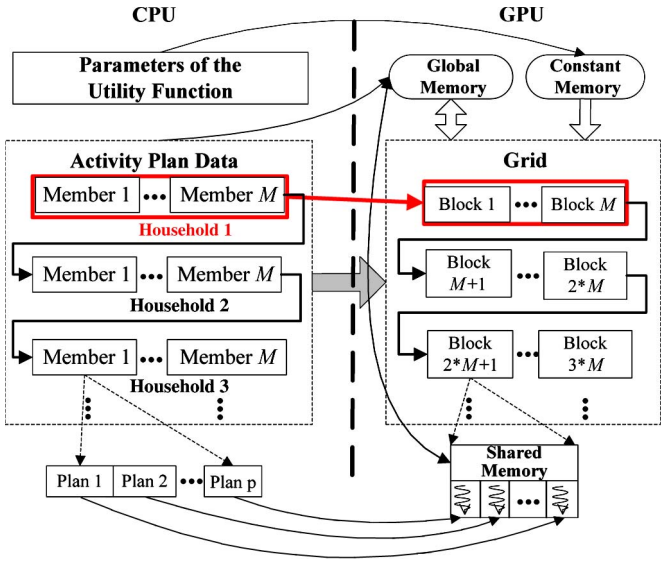

Fig. 5. Data structures and computing resource allocations.

TABLE I
FACILITY LIST

Facility	Opening Time	Location				
		1	2	3	4	5
Home	00:00-24:00	(67,1)	(25,37)	(15,91)	(33,90)	(26,64)
Work	7:00-19:00	(32,67)	(45,73)	(11,48)	(55,2)	(37,99)
School	6:00-18:00	(21,58)	(23,78)	(22,7)	(45,32)	(68,89)
Bar	17:00-24:00	(55,87)	(23,57)	(42,78)	(93,62)	(14,39)
Market	6:00-21:00	(42,11)	(96,12)	(5,60)	(56,76)	(12,19)
Gym	9:00-21:00	(79,34)	(21,90)	(39,85)	(33,21)	(5,98)
Mall	9:00-21:00	(79,34)	(21,90)	(39,85)	(46,39)	(88,14)

IV. EXPERIMENTS

In this section, we set the marginal utility of an activity at its typical duration is $\beta_{\rm dur} = 20$; the coefficient of the activity duration is $c = 200$; the coefficient of traveling is $\beta_{\rm trav} = -12$; the coefficient of waiting is $\beta_{\rm wait} = -6$; the coefficients of arriving late and leaving early are $\beta_{\rm late.ar} = -18$ and $\beta_{\rm early.dp} = -18$, respectively; the coefficient of short duration of the activity is $\beta_{\rm short.dur} = -6$; and $\sigma^2_{t_W} = t_W/10$. The facility list is given out in Table I, which contains the opening times and locations of the facilities.

A. Generate Daily Activity Plans for Individual Agent

In this part, we generate daily activity plans for multiple individual agents, and the activity lists are given in Table II.

As shown in Table II, p is the priority of an activity, t_W is the typical duration, $t_{\rm latest.ar}$ is the latest starting time, $t_{\rm earliest.dp}$ is the earliest ending time, and $t_{\rm shortest.dur}$ is the shortest duration. The time unit of the parameters is hour.

Among the activities, "Sleep," "Breakfast," "Lunch," "Dinner," "Early work," "Late work," "Buy food," and "Leisure at home" are definitely to be performed, whereas the others are optional. For one agent, the locations of home and work are randomly selected in the first generation and will not change in the remaining generations, but the location of the other facilities can be randomly selected in each generation.

TABLE II
DAILY ACTIVITY LIST OF INDIVIDUAL AGENT

Activity	p	t_W	$t_{latest.ar}$	$t_{earliest.dp}$	$t_{shortest.dur}$	Facility
Sleep	1	7.0	23.0	30.0	6.0	Home
Breakfast	2	0.5	8.0	6.0	0.25	Home
Lunch	1	1.25	14.0	12.0	0.75	Work
Dinner	1	1.5	21.0	18.0	0.75	Work
Early work	1	4.0	9.0	11.5	3.0	Work
Late work	1	4.0	15.0	17.0	3.0	Work
Buy food	2	0.5	20.0	18.0	0.25	Market
Drink beer	3	2.0	22.0	18.0	0.5	Bar
Bodybuilding	3	1.5	20.0	18.0	0.5	Gym
Leisure at home	3	1.5	22.0	20.0	0.5	Home

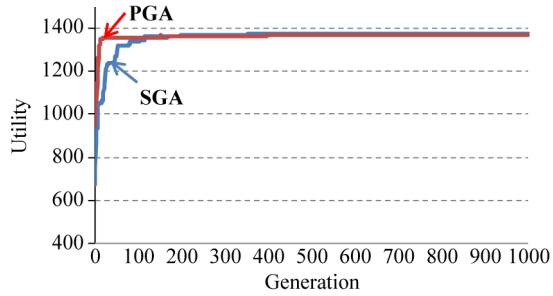

Fig. 6. Results of a typical run of the SGA and the PGA.

In hardware, we use a personal computer with one AMD Athlon 64 X2 dual-core processor 4000+ and an NVIDIA Tesla C2050 GPU, and in software, we use a CUDA driver and SDK with version 3.2. To illustrate the effectiveness of the PGA, we use both the simple genetic algorithm (SGA) proposed by Goldberg [22] and the PGA to generate the daily activity plans. The population sizes of the two algorithms are set as 512, which makes the grid of the GPU consist of 1000 blocks, and one block consists of 512 threads. The number of generations is set as 1000, and the probabilities of the crossover and the mutation are set as $P_c = 0.95$ and $P_m = 0.1$, respectively. For one typical run, the convergence processes of the SGA and the PGA are shown in Fig. 6, and the details of the activity plans after 1000 generations by the SGA and PGA are shown in Table III. In Fig. 6, the PGA converges faster than the SGA. We have done many experiments for the comparison of the two algorithms. There is no obvious evidence which one is better. This is reasonable as the two algorithms are based on the same idea and mechanism.

To reveal the computing power of the GPU, the SGA program is run on the CPU without activating the GPU. The speedup results in Table IV are based on 30 runs for both the PGA and the SGA. The whole process of the PGA takes about 98 s, and it is about 2312 s for the SGA. A speedup by a factor of 23 is obtained.

Furthermore, we make a simple comparison of the SGA model and DCM given in [4]. The result of DCM is shown in Table V.

TABLE III
DAILY ACTIVITY PLAN OF INDIVIDULE AGENT

Activity	Travel time (SGA/PGA)	Execution Time (SGA/PGA)	Location (SGA/PGA)
Breakfast		05:21-05:54/ 05:42-06:18	Home 3/ Home 3
Early work	05:54-06:29/ 06:18-06:53	06:29-11:35/ 06:53-11:30	Work 2/ Work 2
Lunch		11:35-12:57/ 11:30-12:47	Work 2/ Work 2
Late work		12:57-16:59/ 12:47-16:58	Work 2/ Work 2
Dinner		16:59-18:16/ 16:58-18:14	Work 2/ Work 2
Buy food	18:16-18:27/ 18:14-18:25	18:27-18:49/ 18:25-18:49	Market 4/ Market 4
Bodybuilding	18:49-19:07/ 18:49-19:06	19:07-20:28/ 19:06-20:24	Gym 3/ Gym 3
Drink beer	20:28-20:36/ 20:24-20:32	20:36-22:05/ 20:32-22:05	Bar 3/ Bar 3
Leisure at home	22:05-22:35/ 22:05-22:35	22:35-00:15/ 22:35-00:16	Home 3/ Home 3
Sleep		00:15-08:59/ 00:16-08:38	Home 3/ Home 3

TABLE IV
TIME CONSUMPTION

	CPU+GPU (PGA)	CPU only (SGA)	Speedup
Average Time/*s*	97.72	2312.39	23.66
Standard Deviation	0.30	2.52	N/A

TABLE V
DAILY ACTIVITY PLAN GENERATED BY DCM MODEL

Activity	Travel time	Execution Time	Location
Breakfast		06:55-07:29	Home 4
Early work	07:29-08:59	08:59-13:09	Work 4
Lunch		13:09-14:42	Work 4
Late work		14:42-18:22	Work 4
Buy food	18:22-19:38	19:38-20:13	Market 3
Drink beer	20:13-20:31	20:31-22:16	Bar 2
Sleep	22:16-22:50	22:50-06:31	Home 4

We use the utility function (1) to evaluate the plan, and its utility is 1047.93, which is smaller than the utility of the plan in Table III. DCM is a fast and effective heuristic. However, its results are usually not as good as the GA. We further test the DCM and the SGA for 30 independent runs. The mean and standard deviation of the utility are 1060.01 and 79.96 for the DCM, respectively, and are 1373.06 and 4.58 for the SGA, respectively.

B. Generate Daily Activity Plans for Household Agents

In this part, we only use the PGA to generate activity plans for multiple three-person household agents. An agent can be described as a family with a husband, a wife, and a child. Their activity lists are shown in Tables II, VI, and VII.

Among all activities in the lists, "Breakfast" and "Leisure at home" are jointly performed by all the members of a household. "Sleep," "Breakfast," "Lunch," "Dinner," "Early work," "Late work," "Early school," "Late school," and "Leisure at home" are definitely to be performed, whereas the others are optional. "Buy food" is only performed by one member of a household.

TABLE VI
DAILY ACTIVITY LIST OF THE WIFE

Activity	p	t_W	$t_{latest.ar}$	$t_{earliest.dp}$	$t_{shortest.dur}$	Facility
Sleep	1	7.0	23.0	31.0	6.0	Home
Breakfast	2	0.5	8.0	6.0	0.25	Home
Lunch	1	1.25	14.0	12.0	0.75	Work
Dinner	1	1.5	21.0	18.0	0.75	Home
Early work	1	4.0	10.0	12.0	3.5	Work
Late work	1	4.0	15.0	16.0	3.0	Work
Buy food	1	1.0	19.0	16.0	0.5	Market
Bodybuilding	3	1.0	20.0	19.0	0.75	Gym
Shopping	2	2.0	19.5	18.0	1.0	Mall
Leisure at home	3	2.5	22.0	20.0	1.0	Home

TABLE VII
DAILY ACTIVITY LIST OF THE CHILD

Activity	p	t_W	$t_{latest.ar}$	$t_{earliest.dp}$	$t_{shortest.dur}$	Facility
Sleep	1	8.0	23.0	30.5	7.0	Home
Breakfast	2	0.5	8.0	6.0	0.25	Home
Lunch	1	1.25	14.0	12.0	0.75	School
Dinner	1	1.0	21.0	18.0	0.75	Home
Early school	1	4.0	8.5	11.5	3.0	School
Late school	1	4.0	15.0	17.0	3.0	School
Buy food	2	0.5	18.0	17.0	0.25	Market
Homework	2	2.0	22.0	19.0	0.5	Home
Soccer	2	1.0	17.0	16.0	0.5	School
Leisure at home	3	2.0	22.0	20.0	0.5	Home

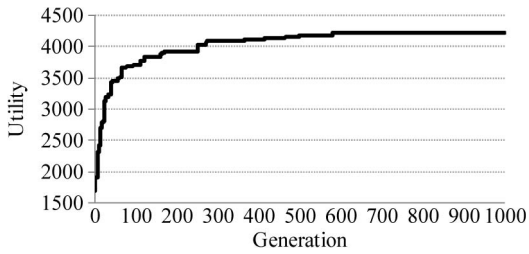

Fig. 7. Results of a typical run of the PGA.

TABLE VIII
DAILY ACTIVITY PLAN OF THE HUSBAND

Activity	Travel time	Execution Time	Location
Breakfast		05:31-05:59	Home 1
Early work	05:59-06:15	06:15-10:19	Work 4
Lunch		10:19-11:42	Work 4
Late work		11:42-16:09	Work 4
Dinner		16:09-17:41	Work 4
Buy food	17:41-17:56	17:56-18:30	Market 1
Bodybuilding	18:30-18:45	18:45-20:15	Gym 4
Leisure at home	20:15-20:54	20:54-22:34	Home 1
Sleep		22:34-06:59	Home 1

Members of one household agent have the same home location, and one member's home and work (or school) locations are constant. For one specific agent, the locations of home and work (or school) are randomly selected in the first generation and will not change in the remaining generations, but the location of the other facilities can be randomly selected in each generation. The parameters are the same with the experiment in Section A. For one typical household agent, the convergence process of the PGA is shown in Fig. 7, and the details of the activity plan after 1000 generations are shown in Tables VIII–X.

TABLE IX
DAILY ACTIVITY PLAN OF THE WIFE

Activity	Travel time	Execution Time	Location
Breakfast		05:31-05:59	Home 1
Early work	05:59-06:14	06:14-10:39	Work 4
Lunch		10:39-11:58	Work 4
Late work		11:58-16:33	Work 4
Shopping	16:33-17:08	17:08-19:19	Mall 5
Dinner	19:19-19:44	19:44-21:20	Home 1
Leisure at home		21:20-00:08	Home 1
Sleep		00:08-08:23	Home 1

TABLE X
DAILY ACTIVITY PLAN OF CHILD

Activity	Travel time	Execution Time	Location
Breakfast		05:31-05:58	Home 1
Early work	05:58-06:44	06:44-11:23	School 3
Lunch		11:23-12:18	School 3
Late work		12:18-16:42	School 3
Soccer		16:42-17:46	School 3
Dinner	17:46-18:32	18:32-19:35	Home 1
Leisure at home		19:35-21:52	Home 1
Homework		21.52-00:03	Home 1
Sleep		00:03-09:20	Home 1

TABLE XI
TIME CONSUMPTION

	CPU+GPU (PGA)	CPU only (SGA)	Speedup
Average Time/*s*	383.28	12,593.49	32.86
Standard Deviation	4.76	21.83	N/A

The speedup results in Table XI are based on 30 runs for both the PGA and SGA. The whole process of the PGA takes about 383 s and is about 12,593 s for the SGA. A speedup by a factor of 32 is obtained.

V. CONCLUSION AND FUTURE RESEARCH

In this paper, we have extended the previous work of using GA to generate daily activity plans to a parallel GA version. Contrary to the previous work of testing GA for a single individual and a single household, we have used the GPU-based PGA to generate activity plans for multiple individual and household agents. We have implemented our algorithm on an NVIDIA Tesla C2050 GPU and have obtained speedup factors of 23 and 32 for 1000 individual and household agents, respectively, compared to the CPU-only implementations. The problem of generating daily activity plans is elementary for the ATSs. In the future, we will go on the research in the three directions.

1) Employ GPU clusters to generate daily activity plans for more agents in parallel.
2) Get more residents' travel data and direct and indirect traffic-related information from real traffic systems to design more reasonable utility functions for the agents.
3) Consider influences of other social relationships such as friends and colleagues and other subsystems such as the weather and legal subsystems on agents' activity planning.

ACKNOWLEDGMENT

The authors would like to thank Dr. F.-H. Zhu for helpful discussions.

REFERENCES

[1] F.-Y. Wang, "Parallel control and management for intelligent transportation systems: Concepts, architectures, and applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 630–638, Sep. 2010.
[2] F.-Y. Wang, "Toward a paradigm shift in social computing: The ACP approach," *IEEE Intell. Syst.*, vol. 22, no. 5, pp. 65–67, Sep./Oct. 2007.
[3] F.-Y. Wang, "Toward a revolution in transportation operations: AI for complex systems," *IEEE Intell. Syst.*, vol. 23, no. 6, pp. 8–13, Nov./Dec. 2008.
[4] F.-H. Zhu, F.-Y. Wang, D. Fan, R. Li, Y. Lv, and S. Chen, "Modeling and analysis of transportation systems using ACP approach," in *Proc. 14th Int. IEEE Annu. Conf. Intell. Transp. Syst.*, Washington, DC, 2011, pp. 2136–2141.
[5] K. Wang and Z. Shen, "Artificial societies and GPU-based cloud computing for intelligent transportation management," *IEEE Intell. Syst.*, vol. 26, no. 4, pp. 22–28, Jul/Aug. 2011.
[6] W. Davidson, R. Donnelly, P. Vovsha, J. Freedman, S. Ruegg, J. Hicks, J. Castiglione, and R. Picado, "Synthesis of first practices and operational research approaches in activity-based travel demand modeling," *Transp. Res. Part A, Policy Pract.*, vol. 41, no. 5, pp. 464–488, Jun. 2007.
[7] F.-Y. Wang and S. Tang, "Concept and framework of artificial transportation system," *J. Complex Syst. Complexity Sci.*, vol. 1, pp. 52–57, Feb. 2004.
[8] G. Xiong and K.-F. Wang, "Parallel traffic management for the 2010 Asian games," *IEEE Intell. Syst.*, vol. 25, no. 3, pp. 81–85, May/Jun. 2010.
[9] N. Zhang and F.-Y. Wang, "DynaCAS: Computational experiments and decision support for ITS," *IEEE Intell. Syst.*, vol. 23, no. 6, pp. 19–23, Nov./Dec. 2008.
[10] H.-X. Zhao, S.-M. Tang, and Y.-S. Lv, "Generating artificial populations for traffic microsimulation," *IEEE Intell. Transp. Syst. Mag.*, vol. 1, no. 3, pp. 22–28, Fall 2009.
[11] F.-H. Zhu, "A case study of evaluating traffic signal control systems using computational experiments," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1220–1226, Dec. 2011.
[12] D. Charypar and K. Nagel, "Generating complete all-day activity plans with genetic algorithms," *Transp.*, vol. 32, no. 4, pp. 369–397, 2005.
[13] K. Meister, M. Frick, and K. W. Axhausen, "Generating daily activity schedules for households using genetic algorithms," in *Proc. 5th Swiss Transp. Res. Conf.*, Monte Verità, Ascona, Switzerland, 2005, pp. 1–25.
[14] K. Meister, M. Frick, and K. W. Axhausen, "A GA-based household scheduler," *Transp.*, vol. 32, no. 5, pp. 473–494, Sep. 2005.
[15] R. Kitamura, "Applications of models of activity behavior for activity based demand forecasting," in *Proc. TMIP Activity-Based Travel Forecast. Conf.*, 1996, pp. 119–150.
[16] T. Arentze, F. Hofman, H. V. Mourik, and H. Timmermans, "ALBATROSS: A multi-agent rule-based model of activity pattern decisions," in *Proc. Transp. Res. Board Annu. Meeting*, Washington, DC, 2000, pp. 136–144.
[17] T. Arentze and H. J. P. Timmermans, "Representing mental maps and cognitive learning in micro simulation models of activity-travel choice dynamics," *Transp.*, vol. 32, no. 4, pp. 321–340, Jul. 2005.
[18] K. Nagel and G. Flötteröd, "Agent-based traffic assignment: going from trips to behavioral travelers," in *Proc. 12th Int. Conf. Travel Behav. Res. Jaipur*, 2009, pp. 1–26.
[19] M. Balmer, N. Cetin, K. Nagel, and B. Raney, "Towards truly agent-based traffic and mobility simulations," in *Proc. 3rd Int. Joint Conf. AAMAS*, New York, 2004, pp. 60–67.
[20] Z. Shen, K. Wang, and F.-H. Zhu, "Agent-based traffic simulation and traffic signal timing optimization with GPU," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst.*, 2011, pp. 145–150.
[21] S. Tsutsui and N. Fujimoto, "Solving quadratic assignment problems by genetic algorithms with GPU computation: A case study," in *Proc. 11th Annu. Conf. Companion Genetic Evol. Comput. Conf.*, Montreal, QC, Canada, 2009, pp. 2523–2530.
[22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.