SOLVING SIGNAL COORDINATION PROBLEMS USING MASTER-SLAVE GENETIC ALGORITHMS

Montty Girianna
Phone (217) 333 5967
girianna@uiuc.edu

Professor Rahim F. Benekohal
Phone (217) 244 6288
Fax (217) 333 1942
rbenekoh@uiuc.edu

Department of Civil and Environmental Engineering
University of Illinois at Urbana Champaign
Newmark Civil Engineering Laboratory
205 N. Mathews Ave.
Urbana, IL 61801

Submitted for publication in the proceeding of the 82nd
Transportation Research Board Annual Meeting
January 2003, Washington, DC.

7,444 words including tables and figures
SOLVING SIGNAL COORDINATION PROBLEMS USING MASTER-SLAVE GENETIC ALGORITHMS
Montty Girianna and Rahim F. Benekohal
University of Illinois at Urbana-Champaign
Abstract

This paper presents the design of master-slave genetic algorithms (GA) in solving signal coordination problems. When a serial GA is applied, its performance in terms of computation time diminishes as more accurate results (smaller time slices to evaluate flows and queues) of network performances are needed, or the size of signal networks increases. Because GA works with a population of independent solutions, it is easy to distribute the computational load, i.e., calculating fitness values of candidate solutions, among several processors and, thus, considerably speed-up the computation time. With a master-slave GA, a single processor (master) performs all genetic operations while a number of processors (slaves) are assigned to evaluate a set of fitness functions. The fundamental step in designing a master-slave GA is to determine the optimal number of processors. In this paper, the analytical formulation of defining the optimal number of processors and the empirical results from the master-slave GA application are presented. A master-slave GA is implemented to a signal coordination problem for a network with oversaturated intersections. For a given network size, the performance of a master-slave GA is investigated when network performances (flows and queues) are evaluated at different sample times, $\Delta T$. When the fitness evaluation time is large (using smaller $\Delta T$) relative to the communication time, the master-slave GA is more efficient and provides larger speed-up. The performance of a master-slave GA is also investigated when the size of signalized networks or the size of problems is changed. Results indicate that an efficient master-slave GA (larger speed-up) is attained when it is used to solve a problem with higher fitness evaluation time and to solve a network with a larger size. Although a master-slave GA only provides a lower bound speed-up, the illustration in this paper demonstrates the potential merit of using parallel GA in solving a signal coordination problem.
INTRODUCTION

In traffic control application, as in many other applications, the search for an optimal solution requires the use of simulation to evaluate the objective or fitness function and to examine constraints. For a large traffic signal network, the simulation takes a significant fraction of optimization time. This computational bottleneck hinders the efficacy of signal coordination, particularly when signal coordination is implemented in real-time. Genetic algorithms (GA) are search techniques and work with a population of independent solutions, which makes it easy to distribute the computational load among several processors and reduces the computation time. The easiest method is to distribute the evaluation of fitness functions among several processors while one master executes the operation of genetic algorithms. This is called a master-slave GA. The fundamental step in designing an efficient master-slave GA is to determine the number of processors. The number of processors used is restricted by the communication time required to exchange data among processors. When a considerable amount of running time is devoted to the data exchange, an efficient master-slave GA is executed with a smaller number of processors. On the other hand, when the data exchange occurs in a smaller fraction of running time, a larger number of processors should be used.

The purpose of this paper is to investigate the performance of master-slave GA when it is used to solve signal coordination problems on a network with oversaturated intersections. Signal coordination is formulated as a dynamic optimization problem. The objective is to maximize the number of vehicles released by a signal network, subject to a set of constraints including ideal offsets, de facto red avoidance, and closed-loop signal timing. To investigate the effect of the evaluation time, a network with $N=20$ signalized intersections is used. To examine the scalability of master-slave GA, five different networks are used, i.e., $N=10$, $N=15$, $N=20$, $N=25$, and $N=30$. The SGI Origin 2000 supercomputing machine is used to execute the master-slave GA. The following section presents the formulation of signal coordination followed by a brief discussion on the fundamental concept of GA and master-slave GA. Next, the performance of the master-slave GA is investigated when flows and queues in a network are evaluated at a different time slice and when the network size is changed. After the results are examined, the paper concludes with findings and recommendations for future works.

SIGNAL COORDINATION MODELS

Optimization problems

This section briefly presents a signal coordination model on oversaturated networks represented as a dynamic optimization problem. The development of signal coordination problems for oversaturated networks is given elsewhere (Gal-Tzur, A., Mahalel, D., & Prashker, N., 1993; Abu-lebdeh & Benekohal, 1997; Lieberman, 2000; Girianna & Benekohal, 2002a). Let $G=(N,L)$ denote a signalized intersection network consisting of a set of signals $N$ and a set of directional streets $L$. Also let $L_p$ be a set of streets along coordinated paths, and $N_p$ is a set of signalized intersections situated on coordinated paths. Equation (1) shows the objective function used in this paper.
Max \[ Z = \sum_{t=1}^{T} \sum_{i,j \in \mathcal{L}} \sum_{h \in H} D_{i,j}^h(t) - \sum_{k} \delta_{i,j}^h(k) q_{i,j}^h(k) \quad \delta_{i,j}^h(k), q_{i,j}^h(k), d_{\text{max}} > 0 \quad (1) \]

\( D_{i,j}^h(t) \) symbolizes departure flows of phase \( h \in H \) at signal \( j \) serving flows from signal \( i \) over a period of \([t \Delta T, (t+1) \Delta T]\). \( H \) is the total phase number, \( \Delta T \) is a sample time interval (say 2, 3, 4, or 5 or more seconds), and \( t=1,2, \ldots, T \) is a discrete time index. \( d_{i,j} \) is the distance from signal \( i \) and \( j \), and \( d_{\text{max}} \) is the maximum length of streets in the network. \( q_{i,j}^h(k) \) is the number of vehicles in queue approaching signal \( j \) coming from signal \( i \) at the beginning of the downstream coordinated green phase, \( h^* \), in cycle \( k \). Depending on the signal plan and traffic movements served by coordinated phase, \( q_{i,j}^h(k) \) may refer to the left-turning or right-turning plus trough movements, or total. \( \delta_{i,j}^h(k) \) is a non-negative disutility factor whose values are determined based on a queue management strategy. In this paper \( \delta_{i,j}^h(k) = 1 \) for \((i,j) \in L_p \) and all \( k \). The factor penalizes the occurrence of queue along coordinated arterials. \( K \) is the period of oversaturation in a cycle number, and \( T \) is the period of saturation in a sample time.

A set of constraints defined by equations (2) to (5) needs to be satisfied for signal coordination. Offsets between signal \( i \) and \( j \), \( \phi_{i,j}^h(k) \), must satisfy equation (2), where \( h \) is coordinated phase. \( \tau_{i,j}(k) \) is the time required for the first vehicle in the released platoon from signal \( i \) to join the tail of the downstream platoon (as the tail has reached its desired speed). \( \alpha_{i,j}(k) \) is the time required for the tail of queue to be released from signal \( j \) to start moving. De facto red due to backed-up traffic on a receiving street must be avoided. This is defined by equation (3), where \( g_i(k) \) is the effective green time for the upstream signal and \( g_j(k) \) is that for the downstream signal. \( \beta_{i,j}(k) \) is the time it takes for a stopping shock wave to propagate upstream.

\[ \phi_{i,j}^h(k) = \tau_{i,j}(k) - \alpha_{i,j}(k) \quad (2) \]
\[ g_i(k) \leq g_j(k) + \phi_{i,j}^h(k) + \beta_{i,j}(k) \quad (3) \]
\[
\sum_{i,j \in F(o)} \phi_{i,j}^h(k) - \sum_{i,j \in R(o)} \phi_{i,j}^h(k) + \sum_{j \in N(o)} g_{j,s}^h(k) = \sum_{m=k}^{k+N} C_{j}^h(m) \quad \forall o \in O \quad (4) \]
\[ q_{i,j}^h(t) \leq \max_{(i,j) \in \mathcal{L}_s} q_{i,j}^h \quad (5) \]
\[ g_{\text{min}} \leq g_i(k) \leq g_{\text{max}} \quad \forall i \in \mathcal{N} \quad (6) \]

The sum of offsets and green times around any loop of the network is equal to an integer multiple of the cycle time (Gartner, 1972). This lock-in constraint is formulated by equation (4). \( N(o) \) is a set of nodes of directed loop \( o \). \( F(o) \) is the set of forward links in loop \( o \), where traffic flow moves in the same direction as the loop’s direction. \( R(o) \) is the set of reverse links in loop \( o \), where traffic flow moves in the opposite direction as loop’s course.
The green time of signal \( j \) at cycle \( k \) serving movements in loop \( o \) is represented as \( g_{j,o}(k) \). \( n_k \) is an integer number indicating the traffic signal cycle number and \( O \) is the number of loops in the network. Queue along non-coordinated arterials must be less than the storage capacity of approach links. This is formulated in equation (5), where \( L_s \) is a set of streets carrying non-coordinated movements. The effective green time must be within a specified lower bound, \( g_{\text{min}} \), and upper bound, \( g_{\text{max}} \), as defined in equation (6).

**Network Loading**

Flows and queues on a signalized intersection network can be evaluated at certain discrete or sample times. The choice of sample times depends on the required accuracy and available computing resources. For a given sample time \( \Delta T \), the number of vehicles departing intersection \( j \) originated from intersection \( i \), \( D_{i,j}^h(t) \), during the effective green for phase \( h \) is expressed in equation (7). \( A_{i,j}^h(t) \) is the arrival flow and depends on the departure flows of upstream intersections feeding street \((i,j)\). The capacity during effective green interval, \( c^h(t) \), equals the saturation flow, \( s^g \) (veh/sec), if the phase \( h \) has a right-of-way. Otherwise, it equals zero. Queue is updated using equation (8).

\[
D_{i,j}^h(t) = \min \left\{ \frac{c^h(t) \Delta T}{A_{i,j}^h(t) + q_{i,j}^h(t-1)} \right\} \quad \forall i, j \in L \tag{7}
\]

\[
q_{i,j}^h(t) = \max \left\{ 0 \left[ q_{i,j}^h(t-1) + \left( A_{i,j}^h(t) - D_{i,j}^h(t) \right) \right] \right\} \tag{8}
\]

**MASTER-SLAVE GENETIC ALGORITHMS (GA)**

**Fundamental of Genetic Algorithms**

Genetic algorithms work with a number of solutions or individuals collectively known as a population. Exploration of the solution space is made by means of exploiting information about the pattern of solutions extracted from the current population. Many classes of genetic algorithms are distinguished according to procedures of the extraction. A simple genetic algorithm (SGA), for example, uses recombination and mutation techniques to extract information (Goldberg, 1989), while Bayesian optimization algorithm (BOA) uses the joint distribution modeled by Bayesian or belief networks to draw out information (Pelikan, 1999). Application of SGA and BOA in signal control problems executed in serial computing machines is given elsewhere (Girianna & Benekohal, 2002b).

In this paper, SGA is executed in a computer with multiple processors. The individual consists of a set of green time (decision) variables, \( g(k) \), where \( k \) is a cycle number. Every variable is first coded in a fixed-length of binary string \( l \), allowing each variable to have \( 2^l \) different possible values. A string consists of genes, each of which has a
To evolve new solutions, an initial population of encoded solutions is created randomly or using some problem-specific knowledge. This population is subjected to genetic operators to evolve to new solutions. Tournament selection (Goldberg, Korb, & Deb, 1989; Goldberg, 1991) with selection pressure $s=2$ is used, in which two individuals are drawn from the population (with or without replacement) and the best individual is selected according to the value of the fitness function (the objective function). This selection scheme provides a similar effect as the truncation selection (Mühlhenbein & Schlierkamp-Voosen, 1993), but provides better performance than the proportionate schemes (Goldberg, 1989; Goldberg, 2002). Crossover is applied to the randomly chosen individual pairs from the population and used to explore or search the space of solutions but preserve the available information stored in the parents’ individuals. In this paper, uniform crossover (Ackley, 1987; Syswerda, 1989) is used in which every allele is exchanged between the two individual pairs with a certain probability $p_e$, known as the swapping probability. Despite being the most disruptive recombination operators, uniform crossover provides a better exploration capability of the solution domains relative to other common crossovers, such as one or two-point crossovers (De Jong, 1991; Eshelman, 1991). The number of individuals in a population ($n$) determines the quality of solutions. As the size of the population increases, the quality of the solutions increases. However, longer computation times are needed for greater population sizes. In this paper, the optimal population size follows the Gambler’s ruin model (Harik, Cantu-Paz, Goldberg, & Miller, 1997, 1999), in which the population size equals the square root of the problem size, i.e., the length of string ($l$).

With selection and crossover operators, and $n=\sqrt{l}$, GA is executed with multiple epochs. Mutation is not used in this research. Its function is replaced by reinitializing the population for every epoch. GA without mutation is also called selectorecombinative GA (Goldberg, 2002). In addition, an elitist strategy is applied where the best individual of the previous generation is kept as one of the current candidate solutions. This type of GA is called Micro GA (Krishnakumar, 1989) and its basic procedure is as follows:

a) Select randomly either a population of size $n$, or $n-1$ and one good individual from any previous search

b) Evaluate fitness and determine the best individual, and carry it to the next generation (elitist strategy)

c) Determine the remaining individuals using genetic operators, i.e., selection and crossover ($p_e=1$)

d) Check for convergence. If the search converges, proceed to step a) (one epoch), otherwise proceed to step b)

Note that even though $p_e=1$ (crossover always occurs), with the elitist strategy, the best solution of a generation is preserved. The process of genetic operation outlined above continues until it reaches the prescribed number of maximum epochs, $e_{max}$. In this research, it is found that with $e_{max}=100$, the solution quality and the computation is acceptable.
Constraints and Fitness functions

Equation (1) needs to be maximized subject to a set of constraints formulated in equations (2) to (6). The range of green time, equation (6), must be within a certain interval, and this requirement can be met by manipulating binary code in GA. A string with all zeros represents the minimum green time, \( g_{\text{min}} \), and a string with all ones represents the maximum green time, \( g_{\text{max}} \). Any other string is decoded to a green time, using \( g_{\text{min}} + DV(g_{\text{max}} - g_{\text{min}})/2^{d-1} \), where \( DV \) is the decoded value of a string. Thus, for \( g_{\text{min}}=20 \) and \( g_{\text{max}}=80 \), the decoded value for a four-bit string 1001 is \((1)(2^3) + (0)(2^2) + (0)(2^1) + (1)(2^0) = 9\), and the corresponding green time for such a string is \( 20 + [(80-20)/(2^4-1)] \times 9 = 66 \) seconds. With this procedure, green time is always within the specified range.

Unlike the constraint for green time intervals, the other constraints require a simulation to check for a violation and may have limits that are solution dependent. In other words, they depend on flows and queues resulted from a network loading. In this paper, GA uses penalties for any violation of the constraints (Goldberg, 1989; Dasgupta and Michalewicz, 1997). The basic procedure is to define the fitness value of an individual \( i \) by extending the domain of the objective function \( Z_i \), i.e., equation (1), using the augmented objective function,

\[
Z_i - \sum_{j} \mu_j H_j
\]

where \( \mu_j \) is a penalty coefficient for constraint \( j \), \( m \) is the number of implicit constraints, and \( H_j \) denotes \( j \)'s constraint function (inequality and equality). \( C_{\text{min}} \) is a coefficient, i.e., the absolute value of the worst possible value of the augmented objective function. This input coefficient is introduced to overcome the negative value of the augmented objective function (Goldberg, 1989), which occurs during early generations of GA as constraints are violated. The augmented objective function becomes a mapping of the objective function to fitness form.

Master-slave Parallelism

The design of parallel GA involves the choice of using one population or multiple populations. The former offers an easier parallel GA design but provides the lower bound speed-up. When the latter is used, while it is potentially the most efficient parallelism, one must accept a larger complexity when deciding how many populations to use and whether the populations remain isolated or communicate by exchanging individuals. This paper uses parallel GA with one population, and assigns a processor (master) to execute genetic operations while multiple processors (slaves) evaluate the fitness function. Four measures of effectiveness (MOE) are used to investigate the performance of master-slave GA: the elapsed time, speed-up, the fitness function evaluation time, and the communication time among processors.

The elapsed time is the time needed for the master-slave GA to solve the entire process of calculation. This calculation includes creating a new generation of candidate solutions (master), distributing the candidate solutions to slaves, evaluation of fitness functions by all processors, and collecting all fitness values from all processors. Unlike slaves, the master takes part in all of the processes. Therefore, the elapsed time per generation taken by the master is recorded as the MOE. The derivation of the theoretical elapsed time per generation described in this research follows the Cantu-Paz approach (Cantu-Paz, 2000). The execution time in parallel GA depends on the time required to evaluate one individual solution (\( T_i \)) and the time required for one processor to communicate.
to each other \((T_c)\). When \(P=S+1\) processors are used (\(S\) is the number of slaves) with \(n\) individuals in the population, the time required for the evaluation is \(E_t=(nT_f)/(P^x)\), where \(x\) is a processor factor quantifying the power of master-slave GA in reducing computation time. Normally \(x=1\). A greater \(x\) indicates a more powerful master-slave. The time required for the communication is \(C_t=(P)(n)(T_c)\). \(T_c\) depends on the amount of information (message) exchanged. For each individual transferred, the communication time can be approximated as \(T_c=B(l_m)/P+L_c\), where \(l_m\) is the length of a message, \(B\) is the inverse of the bandwidth of computer networks, and \(L_c\) is the latency of communication, or overhead, per message. The overhead depends on the operating system, the programming environment, and the hardware. Thus, the elapsed time for one generation of the master-slave GA is \((C_t+E_t)\) and is estimated by equation (9), where \(\alpha_n=B(n)(l_m)\) and \(\beta_n=L_c(n)\). \(\beta_n\), the slope of the equation, represents the additional communication time required when a processor is added in the master-slave GA. This equation shows that as more processors are used, the computation time decreases, but simultaneously, the communication time increases. The net effect depends on the relative magnitude of the evaluation time to the communication time. In general, \(T_c<<T_f\), and the master-slave GA reduces the elapsed time. Note also that the number of processors used in master-slave GA cannot be larger than the number of individuals in a population. When \(p>n\), one or more processors will be idle. Thus, \(P\leq n\).

\[
T_p = (P)(n)(T_c) + \frac{(n)(T_f)}{(P^x)} \\
= (P)(n)\left(B\frac{(l_m)}{P}+L_c\right) + \frac{(n)(T_f)}{(P^x)} \\
= B(n)(l_m) + L_c(n)(P) + \frac{(n)(T_f)}{(P^x)} \\
= \alpha_n + \beta_n(P) + \frac{(n)(T_f)}{(P^x)} \tag{9}
\]

For every generation, communication between a master and slaves takes place two times. The first is when a master broadcasts a set of candidate solutions (individuals) to all slaves. During this communication, the master sends a copy of solutions (a set of green time intervals coded in binary numbers) to all slaves along with instructions as how to evaluate individuals. When the ratio of the population size to the number of processors used \((n/P)\) is an integer number, all processors (the master and slaves) evaluate the same number of individuals. For example, if \(n=10\) and \(P=5\), then the ratio \((n/P)=2\), and each processor evaluates two different individuals. When the ratio is not an integer number, some slaves evaluate more individuals than others. The second communication between the master and slaves occurs when the master collects all the fitness values sent from the slaves that have completed the evaluation. The time required to execute these two communication processes is defined as the communication time \(T_c\) introduced earlier and is recorded for each generation. The length of a message \((l_m)\), as used in equation (9), is the average size of the message exchanged during the first and the second communication.
The time spent to evaluate fitness functions, $E_t$, depends on the population size, $n$, and the number of processors used, $p$. Depending on the number of individuals evaluated by each processor, the duration of evaluation may not be identical for all slaves. When $n/p$ is not an integer, one processor may evaluate a greater number of individuals as compared to others and, thus, takes a longer time to finish the evaluation. Together, the master and slaves evaluate fitness values. Before proceeding to the next generation, the master has to wait until all slaves have completed the evaluation and it has received all the fitness values of all candidate solutions. The evaluation time is recorded as the master receives the last message.

To obtain the optimal number of processors, one sets the first derivative of equation (9) equal to zero and solves for $P$. With $x=1$, the number of optimal processors is determined by equation (10), where $\gamma = (T_f)/(\beta_n)$. Note that for master-slave GA, $P^* \leq n$.

$$P^* = \sqrt{(n)T_f} / \beta_n = \sqrt{(n)(\gamma)}$$

(10)

To ensure that the master-slave PGA provides an efficient run, equation (11) must hold, where $T_s$ is the elapsed time for serial GA. The ratio $T_s/T_p$ is called the parallel speed-up of the master-slave GA.

$$\frac{T_s}{T_p} = \frac{(n)T_f}{\alpha_n + \beta_n P + \frac{(n)T_f}{P^*}} = \frac{(n)\gamma}{\alpha_n + P + \frac{(n)\gamma}{P^*}} > 1$$

(11)

Solving for $\gamma$ and assuming $x=1$ results in the necessary condition that the master-slave GA performs better than serial GA.

$$\gamma > \frac{\alpha_n(P)}{\beta_n(n(1-1/P))}$$

(12)

Another parameter used to measure the performance of master-slave GA is the efficiency of using more processors. The efficiency of master-slave GA, $E$, is formulated below.

$$E = \frac{T_s}{T_pP}$$

(13)

Ideally, $E$ equals one, or $T_s=(T_p)(P)$, and the parallel speed-up, $T_s/T_p$, equals the number of processors, i.e., linear speed-up. In practice, the ideal conditions may not occur due to the cost of communication among processors and whether or not the master-slave GA
fully utilizes the processors. The efficiency, \( E \), measures the deviation from the ideal conditions.

**MPI and SGI Origin 2000**

In this paper, the Message-Passing Interface (MPI) is used to perform the data transfer between processors. MPI is a library of functions and macros that can be used in C, FORTRAN, and C++ languages. In this research, a broadcasting concept of data transfer is used, that is, a collective communication in which a single processor (master) sends the same data to every processor. To collect the fitness values from all processors, a collective reduction operation is used. With this operation, the master has access to all slaves and collects the fitness values from them. With these two data transfer functions (broadcast and reduction operation), the master-slave GA evaluates all individuals defined in a set of candidate solutions. For each generation, all processors use these two MPI procedures. The SGI Origin 2000 of the National Center for Supercomputing Applications (NCSA) is used to execute the master-slave GA. This machine is a scalable shared memory multiprocessor (S2MP), also known as a distributed shared memory machine. Its memory is physically distributed but virtually shared.

**THE EFFECT OF EVALUATION TIMES**

This section presents the effect of the duration of evaluation time on the efficacy of the master-slave GA. If the evaluation time is small relative to the communication time, then the efficacy of the strategy may be hindered. On the other hand, when the evaluation time is large relative to the communication time, the strategy works efficiently. Two different cases are examined. The first case is when flows and queues are evaluated at 10-second intervals (Case 1) and the second is when they are evaluated at five-second intervals (Case 2). Thus, the evaluation time for Case 1 is expected to be smaller than that for Case 2. A network of arterials \( G=(N,L) \), consisting of a set of signals \( N=20 \) and a set of directional (one-way) streets \( L=49 \) (one-way including exit and entry links), as shown in Figure 1, is used as an example. Each signal works with a two-phase plan, \( h=2 \). Signal coordination is made for the northbound arterial from signal 20 to 5 and for those along arterials running E-W. The thick lines in the figure display the coordinated movements. Decision variables are green times for 15-minute periods of oversaturation, or approximately 15 cycles. All of the decision variables have to be represented in GA individuals. Thus, for a network with \( N=20 \) signals evaluated for \( K=15 \) cycles, the decision variables are \((h)(N)(K)=(2)(20)(15)=600\) variables. In this paper, each variable is represented as four bits of binary number, and the length of GA individual becomes \( l=(4)(600)=2,400 \) and \( n=50 \).

**Load distribution among slaves**

For each case, one processor is first used. With an increment of one processor, the case is subsequently executed until the number of processors or slaves equals the population size \((n/p)\). For example, if population size \( n=36 \), the master-slave GA is executed using \( p=2, 3, 4 \), and so on until \( p=36 \) processors. For a given population size, the load balancing among slaves requires that all slaves and the master evaluate the same number of individuals. If \((n/p)\) is an integer, both the master and a slave each evaluate \((n/p)\) individuals for each generation, i.e., balanced-load conditions. When \((n/p)\) is not an integer, some processors
evaluate as many as \((n/p)\) individuals plus one, while others evaluate only \((n/p)\), i.e., unbalanced-load conditions. Hence, for \(n=30\) and \(p=12\), some processors evaluate three individuals while the remaining processors evaluate only two individuals. For each case and for a given number of processors, ten runs are made with each different seed number used by master-slave GA for generating candidate solutions. For each run, the average evaluation time, \(E_t\), and the communication time per generation, \(C_t\), are recorded.

**Elapsed time and speed-up**

Figure 2(a) shows the average elapsed time per generation, and Figure 2(b) shows the speed-up for both cases. The two measures are represented as a function of the number of processors used. As expected, for a given number of processors, the case with 10-second sample times (Case 1) requires less time for GA to evaluate. When two processors are used, the elapsed time for Case 1 decreases from 3.369 seconds (one processor) to 1.722 seconds, i.e., the speed-up is approximately two. Adding more processors provides less elapsed time and increased speed-up. A linear speed-up is maintained approximately until \(p=5\). When the number of processors equals the population size, i.e., \(p=n\), the maximum speed-up is attained. Note that the master-slave GA is more effective when the balanced-load conditions occur (shown as dashed curves in Figure 2(b)). For example, with one processor evaluating two individuals, i.e., \(p=25\) (balanced-load conditions), the speed-up does not increase by adding more processors until \(p=50\). This is because for \(25<p<50\), at least one processor has to evaluate two individuals, and this bottleneck determines the duration of the entire computation time. For Case 2 (five-second sample times), a linear speed-up is maintained approximately until \(p=10\). Similar to Case 1, the master-slave GA is effective when balanced-load conditions occur, and the maximum speed-up is attained when \(p=n\). The maximum speed-up for Case 2, i.e., 28, is larger than that for Case 1, i.e., 22.

**Optimal number of processors**

Equation (10) determines the optimal number of processors. To determine \(T_f\) and \(\beta_n\) for each case, a regression analysis is performed. Figure 3 (a) shows the recorded \(C_t\) and its regression line for both cases. For a case with 10-second sample times, it is found that the communication time is expressed as \(C_t=(8.657)+(0.691)(p)\). The slope of this regression line represents the additional communication time required for an additional processor used in master-slave GA (\(\beta_n\)). Thus, for Case 1, \(\beta_n=0.691\) milliseconds. For Case 2 (10-second sample times) the relationship between \(C_t\) and the number processors is expressed as \(C_t=(9.785)+(0.677)(p)\), with \(\beta_n=0.677\) milliseconds. Notice that \(\beta_n\)'s for both cases are approximately the same. This is expected since the size of messages (the length and the number of individuals in the GA population) being communicated does not change from Case 1 to Case 2.

To determine \(T_f\), recall that \(E_t=(n)(T_f)/(p^x)\). Take a logarithmic function of the equation to obtain a regression line of log \((E_t)\) as a function of log \((p)\) and, thus, log \((E_t)=\log(n)(T_f)-(x)(\log(p))\). Figure 3 (b) reveals the recorded \(E_t\) and the regression line for both cases. The regression line follows log \((E_t)=(7.874)-(0.828)(\log(p))\) for the Case 1 and log \((E_t)=(8.502)-(0.826)(\log(p))\) for Case 2. Therefore, for Case 1, with \(n=50\), \(T_f=(e^{7.874}/50) = 52.55\) milliseconds, the processor factor \(x=0.828\), and \(\gamma=T_f/\beta_n=(52.55)/(0.691) = 76\). With \(\gamma\) known, the optimal number of processors is \(p^x=[(x)(n)(\gamma)]^{1/(1+x)} = [(0.828)(50)(76)]^{1/1.828} =\)
82 processors. With a similar procedure, \( T_f = 98.53 \) milliseconds for a case with five-second sample times, and the optimal number of processors \( p^* = 118 \). As expected, \( T_f \) for a case with five-second sample times is approximately twice as large as \( T_f \) for a case with 10-second sample times.

The (theoretical) optimal number of processors \( p^* \) obtained using equation (10) for both cases is larger than the number of population. This can be observed from Figure 2(b) as the speed-up curves for balanced-load conditions do not yet attain their peak at \( p=n \). If \( p \) were allowed to exceed \( n \), the curves can be extrapolated to reach the theoretical maximum speed-up. Master-slave GA, however, cannot use the (theoretical) optimal number of processors that is larger than the number of individuals in the population. At most, one processor evaluates a single individual. With this constraint, for the two cases previously demonstrated, the number of processors that provides the highest speed-up equals the population (\( n=50 \)), i.e., \( p^*=50 \). This outcome, \( p^*=n \), cannot be applied into general cases because it depends on the magnitude of communication time relative to the evaluation time. According to equation (10), when \( \gamma \) is large, or \( T_c << T_f \), \( p^* \) tends to be larger than \( n \), and one has to design master-slave GA with \( p^*=n \). For practical purposes, the number of optimal processors follows equation (14).

\[
p^* = \begin{cases} 
\sqrt{(n)(\gamma)} & \text{if } \sqrt{(n)(\gamma)} \leq n \\
n & \text{if } \sqrt{(n)(\gamma)} > n 
\end{cases}
\]  

(14)

Efficiency

Efficiency, as defined in equation (13), measures the deviation of master-slave GA’s performance from the ideal conditions (linear speed-up). Communication costs, \( C_r \), and the degree of utilization of processors determine the master-slave GA efficiency. When the number of processors used is small, the communication cost, \( C_r \), is considerably small compared to evaluation time, \( E_t \), and, thus, the elapsed time is greatly determined by \( E_t \). However, as a larger number of processors is used, the gap between \( C_r \) and \( E_t \) becomes closer and both \( C_r \) and \( E_t \) determine the elapsed time. When \( p=3 \) for a case with 10-second sample times, for example, \( E_t = 1,135.0 \) milliseconds and \( C_r=9.8 \) milliseconds, or \( E_t/C_r=116 \), i.e., the evaluation time is 116 times larger than the communication time. When \( p=30 \), \( E_t = 139.0 \) milliseconds and \( C_r=25.5 \) milliseconds, or \( E_t/C_r=5.4 \). With more processors, the gap between \( E_t \) and \( C_r \) becomes smaller. Clearly, the effect of \( C_r \) on the elapsed time becomes significant as a larger number of processors used. In other words, as \( p \) increases, a larger fraction of time is devoted to exchange the data among processors, decreasing the efficiency. Moreover, when a larger portion of processors are idle, \( E \) drops very quickly. This occurs when the master-slave GA is executed with unbalanced-load conditions. When \( p=25 \) (balanced-load conditions), the speed-up for a case with five-second sample times (Case 2) is 17.4 and the associated efficiency \( E=(T_s/T_p)(1/p)=(17.4)(1/25) =69\% \). When \( p=36 \) (unbalanced-load conditions), the speed-up only increases to 18, but \( E \) drops to 50\% because a larger fraction of idle time occurs. In other words, \( p=25 \) is more efficient than \( p=36 \). When \( p \) further increases to 50 (balanced-load conditions), the speed-up is 28 and \( E \) increases to...
56%. Accordingly, the merit of master-slave GA is more noticeable for balanced-load conditions.

**EFFECT OF A NETWORK SIZE**

This section investigates the performance of master-slave GA when the size of networks is changed. The investigation begins with a small one-way arterial network with \( N=10 \) signalized intersections. Five additional intersections are made to construct a new and larger network. The additional signals serve additional arterial for east/westbound direction. The increment is made until \( N=30 \) signals. In other words, the investigation is based on five cases, i.e., \( N=10, N=15, N=20, N=25, \) and \( N=30 \) signalized intersections. Figure 1 shows the network when \( N=20 \). Signals in a network work with a two-phase plan. Similar to a case previously presented, decision variables are green times for 15 minute-periods of oversaturation, or approximately \( K=15 \) cycles. Thus, for a network with 20 signals and for 15 cycles, the length of GA individual is \( l=2,400 \) variables and the optimal population size is \( n=50 \). Flows and queues are evaluated at a sample time of 10 seconds. Table 1 (a) summarizes the network size, variables to be solved, and the optimal number of GA population size for all network sizes. For each case, a similar procedure of load distribution among slaves described earlier is used.

**Elapsed time and speed-up**

Figure 4 (a) shows results of the investigation measured in speed-up. For all cases, a linear speed-up is achieved when \( p<10 \) and deviation from the ideal conditions appears for a larger number of processors. For all possible \( p \), higher speed-up is observed for a case with larger networks, and the number of processors that provides the highest speed-up is \( p=n \). Figure 4 (b) shows the elapsed time for all cases with each curve corresponding to a fixed number of processors. When one processor is used, the elapsed time per generation that is required to solve 10-signal network is about one second. Doubling the network size triples the elapsed time. \( T_f=1.06 \) seconds for \( N=10 \), and it becomes 3.23 seconds for \( N=20 \). The slope is about \((3.23-1.06)/(20-10) = 0.217\), or about two additional seconds for every additional 10 signals. The slope is even higher for larger networks. When multiple processors are used, the slope is less than 0.217. When two processors are used, the slope is about 0.113, or approximately one second for every additional 10 signals. For a larger number of processors, the slope becomes flatter and the increase in a network size requires only a smaller increment in elapsed time.

**Optimal number of processors**

A similar procedure as presented earlier is used to determine the optimal number processors. The results are shown in Table 1 (b). From the table, it can be seen that \( T_f \) increases as the size of network increases. Thus, for example, \( T_f = 24.67 \) milliseconds for \( N=10 \). It increases to \( T_f = 52.55 \) milliseconds (2 times) as the network size is doubled \((N=20)\), and to \( T_f = 76.94 \) milliseconds \((3.1 \) times\) when the network size is further increased three times larger \((N=30)\). Similarly, the slope of communication time increases from \( \beta_n=0.406 \) milliseconds to \( \beta_n=0.921 \) milliseconds when the network size increases from \( N=10 \) to \( N=30 \). The range of the (theoretical) optimal number of processors is from \( p^*=61 \) to \( p^*=97 \) processors, from the smallest to the largest networks. Notice that \( p^* \) is larger than the population size \( n \) for all cases. For the implementation of master-slave GA, \( p^* \) is chosen using equation (14), i.e.,
Thus, for example, as shown in Figure 5 (b), the maximum speed-up occurs when $p=36$ for $n=36$ ($N=10$), therefore $p^*=36$. Similarly, $p^*=44$ for $n=44$ ($N=25$), $p^*=50$ for $n=50$ ($N=20$), etc.

**CONCLUSIONS AND RECOMMENDATIONS**

This paper presented the design of master-slave GA in solving signal coordination problem. Master-slave GA performed better when a smaller fraction of running time is devoted to communication costs. It is well suited to signal coordination problems when a higher accuracy in evaluating flows and queues is demanded, and when it is used to solve a larger network size. The results, however, provided a lower bound on the potential speed-up of parallel GA. Moreover, for a larger number of processors, master-slave GA performs better when the evaluation of fitness function is uniformly distributed among processors. Parallel GA that can use the available processing power more efficiently, but provide, at least, the same quality solution, need to be explored. An alternative to using a single master processor for an entire population is to distribute the population to several processors (multiple populations), and all processors perform genetic operations as well as evaluate fitness functions. Finally, different techniques of evolving solutions, such as using Bayesian network model and executed in parallel, need to be examined.

**REFERENCES**


LIST OF TABLES

1. FIGURE 1 One-way arterial network with 20 signals
2. FIGURE 2 Elapsed time and speed-up per generation of master-slave GA for cycle and discrete-time based signal coordination models
3. FIGURE 3 (a) Communication time, $C_t$, for the cycle-based model ($n=36$) and the discrete-time model ($n=50$)
4. FIGURE 3 (b) Evaluation time, $E_t$, for the cycle-based model ($n=36$) and the discrete-time models ($n=50$)
5. FIGURE 4 (a) Effect of network size on the master-slave SGA elapsed time per generation
6. FIGURE 4 (b) Effect of network size on the master-slave SGA speed-up per generation
7. FIGURE 5 (a) Elapsed time as the network size increases
8. FIGURE 5 (b) Evaluation ($E_t$) and communication times ($C_t$): network with 20 signals
LIST OF FIGURES

1. TABLE 1 (a) Network size and decision variables
2. TABLE 1 (b) Optimal number of processors for master-slave GA
FIGURE 1 One-way arterial network with 20 signals
FIGURE 2(a) Elapsed time per generation of master-slave GA

FIGURE 2(b) Speed-up per generation of master-slave GA
FIGURE 3 (a) Communication time, $C_t$, for $\Delta T=5$ and $\Delta T=10$ seconds ($N=20$)

FIGURE 3 (b) Evaluation time, $E_t$, for $\Delta T=5$ and $\Delta T=10$ seconds ($N=20$)
FIGURE 4 (a) Speed-up per generation for different size of networks

FIGURE 4 (b) Elapsed time for different $p$ with network size increases
TABLE 1 (a) Network size and decision variables

<table>
<thead>
<tr>
<th>No</th>
<th>No. of Signalized Intersections</th>
<th>Individual Size</th>
<th>Optimal Population Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>(2)(10)(15)(4) = 1,200</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>(2)(15)(15)(4) = 1,800</td>
<td>44</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>(2)(20)(15)(4) = 2,400</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td>(2)(25)(15)(4) = 3,000</td>
<td>56</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>(2)(30)(15)(4) = 3,600</td>
<td>60</td>
</tr>
</tbody>
</table>

TABLE 1 (b) Optimal number of processors for master-slave GA

<table>
<thead>
<tr>
<th>No. of Signalized Intersections</th>
<th>Individual Size</th>
<th>Optimal Population Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (2)(10)(15)(4) = 1,200</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>2 (2)(15)(15)(4) = 1,800</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>3 (2)(20)(15)(4) = 2,400</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>4 (2)(25)(15)(4) = 3,000</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>5 (2)(30)(15)(4) = 3,600</td>
<td>60</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network size (N)</th>
<th>N=10</th>
<th>N=15</th>
<th>N=20</th>
<th>N=25</th>
<th>N=30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size, n</td>
<td>36</td>
<td>44</td>
<td>50</td>
<td>56</td>
<td>60</td>
</tr>
<tr>
<td>Evaluation time (milliseconds), $T_f$</td>
<td>24.67</td>
<td>37.90</td>
<td>52.55</td>
<td>64.20</td>
<td>76.94</td>
</tr>
<tr>
<td>Slope of the communication time, $\beta_n$</td>
<td>0.4054</td>
<td>0.5128</td>
<td>0.6909</td>
<td>0.8136</td>
<td>0.9205</td>
</tr>
<tr>
<td>$T_f/\beta_n$</td>
<td>60.85</td>
<td>73.91</td>
<td>76.07</td>
<td>78.90</td>
<td>83.59</td>
</tr>
<tr>
<td>Processor factor, $x$</td>
<td>0.8190</td>
<td>0.8202</td>
<td>0.8284</td>
<td>0.8236</td>
<td>0.8208</td>
</tr>
<tr>
<td>Optimal number of processors, $p^*$</td>
<td>61</td>
<td>76</td>
<td>82</td>
<td>90</td>
<td>97</td>
</tr>
</tbody>
</table>