社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－ARC－182 (27)
2009－HPC－119 (27)
2009／2／28

Parameter-less GA based Crop Parameter Assimilation with High Performance Computing

KEIGO SAKAMOTO,[†1] SHAMIM AKHTER[†1] and KENTO AIDA[†2,†1]

Country level agricultural activities monitoring is necessary to ensure the food security. SWAP-DLGA (SWAP-Double Layer GA) is a simulation model to predict the agricultural information from satellite images. This paper proposes a method to run SWAP-DLGA with an automatic generation technique of suitable GA parameters. The proposed method computes the suitable GA parameters using the Parameter-less GA (PLGA) and runs SWAP-DLGA with the computed GA parameters. Both SWAP-DLGA and PLGA require huge computational time. This paper also presents implementation of the proposed method on a distributed computing system.

1. Introduction

Monitoring of country level agricultural activities is now necessary to ensure the food security problem. The monitoring data, such as satellite images, can be used to estimate growth of agricultural products by using sophisticated estimation models proposed in the agricultural community, e.g. the SWAP (Soil, Water, Atmosphere, Plant) model[1]. However, satellite images do not provide complete information, or crop parameters, required by the sophisticated estimation model.

The SWAP Double Layer GA (SWAP-DLGA)[4] is one of methods to solve this problem. SWAP-DLGA assimilates missing crop parameters in satellite images by running the genetic algorithm (GA). A problem in SWAP-DLGA is that the user needs to define suitable parameters for running GA, or GA parameters such as population size and maximum generation, in advance. However, the suitable GA parameter needs to be found empirical way, and it is not realistic assumption that the user in the agricultural community can find the suitable GA parameters.

This paper proposes a method to run SWAP-DLGA with an automatic generation technique of suitable GA parameters. The proposed method computes the suitable GA parameters using the Parameter-less GA (PLGA)[5] and runs SWAP-DLGA with the computed GA parameters. Both SWAP-DLGA and PLGA require huge computational time. This paper also presents implementation of the proposed method on a distributed computing system, or a cluster of clusters, to reduce the computation time. The implementation includes a load balancing technique in the application level.

†1 Tokyo Institute of Technology
†2 National Institute of Informatics

2. Background

SWAP-GA model[2] is an assimilation technique for missing crop parameters in satellite images. It has been used for estimating soil hydraulic functions and quantifying irrigation characteristics. The parallelization method of SWAP-GA with the master-worker model[3] was proposed and implemented on cluster and grid computing systems. SWAP-DLGA[4], which is extended from SWAP-GA, was proposed. The idea of SWAP-DLGA is to fusion two kinds of satellite images, daily obtainable Low Resolution (LR) images and not daily obtainable High Resolution (HR) images. SWAP-DLGA performs two GA procedures in a hierarchical way, and selecting suitable parameters for GAs is very important issue here.

Usually, it is difficult to find out suitable GA parameters without empirical way. And most users are not interested in GA parameters but in results. The parameter-less GA (PLGA) is proposed to meet the above request.

The rest of this section outlines the techniques organizing the proposed method.

2.1 SWAP-DLGA

The idea of SWAP-DLGA is assimilating missing crop parameters from two kinds of satellite images, HR and LR, by running GAs in a hierarchical manner. This paper uses the terms "outside GA" and "inside GA" to refer the GAs. One individual of the outside GA has missing crop parameters, which are defined in **Table 1**. GWJan and GWDec mean

Table 1 The individual in the outside GA

Gene Number	Lower Boundary	Upper Boundary	Units	Description
1	10	200	cm	GWJan
2	10	200	cm	GWDec
3	1	160	DOY	DEC
4	1	120	DOY	STS
5	90	150	day	TEC

groundwater depth in Jan 1^{st} and Dec 31^{st}, respectively. DEC and STS indicate the date of emergence or crop seeding and date of starting irrigation, respectively. TEC means time extent of crop or the total time that crops are in field, and DOY is day of year. The evapotranspiration (ETa), which is the combination of soil evaporation and plant transpiration, is obtained both from satellite images (SatETa) and the SWAP model with the above crop parameters (SimETa). The difference between SatETa and SimETa is used to evaluate the fitness.

The cost in one pixel of HR C_{xyHR} is defined by eq(1) (Difference1 in **Fig. 1**).

$$C_{xyHR} = \frac{1}{q}\sum_{d}^{d_q}{}_{d_1}(ETa_{xyHR,d} - ETa_{SWAPxyHR,d})^2 \quad (1)$$

$ETa_{xyHR,d}$ and $ETa_{SWAPxyHR,d}$ denote SatETa and SimETa obtained from a HR image of the location indicated by the coordinate (x, y) and the date $d \in [d_1, ..., d_q]$, respectively.

The cost in LR C_{LR} is defined by eq(2) (Difference2 in Fig. 1).

$$C_{LR} = \frac{1}{n}\sum_{D}^{D_n}{}_{D_1} \times [ETa_{LR,D} - \frac{1}{p^2}\sum_{xy}^{x_p y_p}{}_{x_1 y_1}(ETa_{SWAPxyHR,D})] \quad (2)$$

$ETa_{LR,D}$ indicates SatETa obtained from a LR image of the date, $D \in [D_1, ..., D_n]$. p is the number of rows or the number of columns in one HR image. A HR image contains p^2 pixels for an area that presented by one pixel in a LR image. Thus, $\frac{1}{p^2}\sum_{xy}^{x_p y_p}{}_{x_1 y_1}(ETa_{SWAPxyHR,D})$ computes the average SimETa of p^2 HR images.

The total fitness, F, is calculated in eq(3).

$$F = \frac{1}{C_{LR} + \frac{1}{p^2}\sum_{xy}^{x_p y_p}{}_{x_1 y_1} C_{xyHR}} \quad (3)$$

For each HR pixel, one GA is created (the outside GA). The outside GA calculates eq(1). However, the problem comes when eq (2) is calculated. Although SimETa for LR (right middle in Fig. 1) contains p^2 pixels, each of which is presented by N individuals, average ETa (right bottom in Fig. 1) contains one pixel presented by p^2 individual. Thus, we need to select p^2 individuals (in Average ETa) from $N \times p^2$ individuals (in SimETa). The inside GA is created to solve this problem. The inside GA calculates eq(2) and eq(3) to get the fitness of all individuals. The outside GA recreates individuals using GA operators based on the fitness. Good individuals, those have higher fitness, are selected for the next generation. The above process of double layer GA is shown in **Fig. 2**.

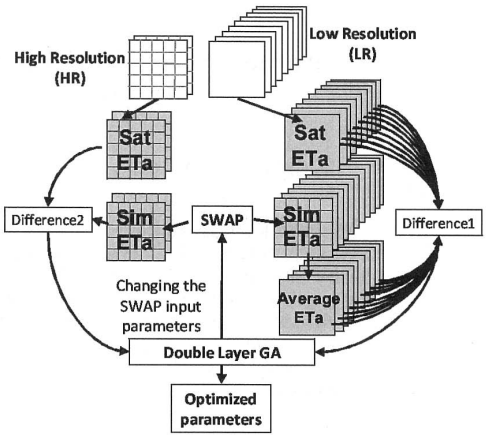

Fig. 1 Overview of SWAP-DLGA

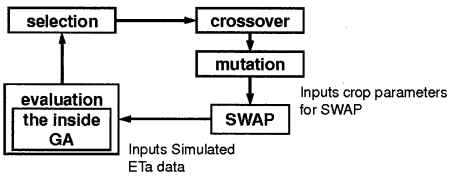

Fig. 2 The process flow of Double Layer GA

2.2 Parameter-less GA (PLGA)

The idea of PLGA is searching suitable GA parameters using some predefined rules. PLGA works well without concerning the setting of suitable GA parameters. The selection pressure s and the crossover probability P_c are preset to fixed effective values ($s = 4, P_c = 0.5$) and the mutation operator is ignored to ensure the convergence of populations.

There is a tradeoff between the population size and computation time. Running GA with smaller population size converges faster than

Table 2 Example of the m-array counter algorithm

Counter (base m=2)	most significant degit changed	Action
0		
1	1	run 1 generation of G_1
10	2	run 1 generation of G_2
11	1	run 1 generation of G_1
100	3	run 1 generation of G_3
101	1	run 1 generation of G_1
110	2	run 1 generation of G_2
111	1	run 1 generation of G_1
1000	4	run 1 generation of G_4
⋮	⋮	⋮

that with larger population size. However, quality of solution with a smaller population size is generally not satisfying. Conversely running GA with larger population size needs more computational time. PLGA in the proposed method uses the followings rule based on the above tradeoff with m-array counter[5] (m is a positive integer and m>2), which is shown in **Table 2**.

- If populaiton size of G_i is N_i, population size of G_{i+1} is set $2N_i$.
  (G is the group of GA processes using same GA parameters and i is the indicator of G)
- At each time step, the counter is incremented.
- The position of the most significant digit that changed during the increment operation indicates which G should be run.
- If $F_{avg,G_i} > F_{avg,G_{i+1}}$, G_i is deleted.
  (F_{avg,G_i} is the average fitness of G_i)

PLGA updates population size from the above rule.

2.3 Distributed SWAP-GA

The paper[3] presents the implementation of the SWAP-GA application on the Grid and discusses the impact on the performance of parallelization methods. It also implemented three parallelization methods, pixel distribution, population distribution, and hierarchical distribution, and compared the performances through the experiments on the Grid testbed. These methods use GridRPC as the programming framework but ways of task distribution are different. Our proposed parallel scheme uses the hierarchical distribution method for SWAP-DLGA with PLGA where SWAP-DLGA is distributed by using the population distribution method. MPI programming framework has been used to distribute the all tasks.

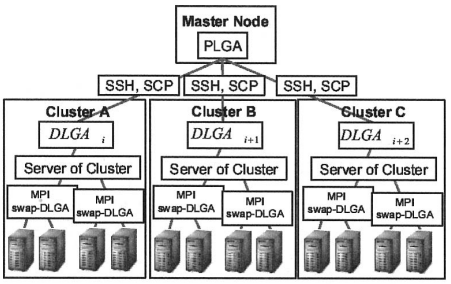

Fig. 3 Implementation scheme on DCS

3. Design and Implementation of SWAP-DLGA with PLGA

This section presents design and implementation of the proposed method. The idea of the proposed method is: (1) running PLGA to select suitable GA parameters, and (2) running SWAP-DLGA with the computed GA parameters. Also, the proposed method runs both PLGA and SWAP-DLGA in parallel on a distributed computing system, or a cluster of clusters. Two questions arise:

(1) Which GA parameter should PLGA compute?
(2) How should both PLGA and SWAP-DLGA be parallelized?

3.1 GA parameters

For the question (1), we conduct preliminary experiments to see effects of GA parameters on performance of the outside GA and the inside GA. The results show that the performance of the inside GA is not significantly affected by selection of GA parameters. So the GA parameters in the inside GA are set fixed values and we run PLGA to find suitable GA parameters for the outside GA in SWAP-DLGA.

3.2 Parallelization

For the question (2), SWAP-DLGA is parallelized in a hierarchical manner on a cluster of clusters. SWAP-DLGAs with different GA parameters are distributed among clusters, where all SWAP-DLGA processes running in a cluster runs with the same GA parameters. Inside the cluster, SWAP-DLGA is parallelized with the population distribution model (see Section 2.3).

We implemented the proposed method on the distributed computing system illustrated in **Fig. 3**. Multiple processes (three processes in Fig. 3) for the outside GA in SWAP-DLGA with different population sizes ($DLGA_x$, x=i, i+1, i+2) are run simultaneously. One clus-

Table 3 The comparision cases and decissions

Comparison	Decision
$F_{avg,DLGA_i} >$ $F_{avg,DLGA_{i+1}} >$ $F_{avg,DLGA_{i+2}}$	No Deletion. Continuously run all the SWAP-DLGAs again with their previous assigned generation number.
$F_{avg,DLGA_i} >$ $F_{avg,DLGA_{i+1}} <$ $F_{avg,DLGA_{i+2}}$ or $F_{avg,DLGA_i} <$ $F_{avg,DLGA_{i+1}} <$ $F_{avg,DLGA_{i+2}}$	Remove $DLGA_i$ and $DLGA_{i+1}$. Additionally create two new SWAP-DLGAs as $DLGA_{i+3}$, $DLGA_{i+4}$ with 2^{i+2} and 2^{i+3} population size correspondingly.
$F_{avg,DLGA_i} >$ $F_{avg,DLGA_{i+1}} <$ $F_{avg,DLGA_{i+2}}$	$DLGA_i$ is deleted and create $DLGA_{i+3}$ with 2^{i+2} population size

ter is assigned to one $DLGA_x$. First, a master node sends the input files for SWAP-DLGAs and runs the slave program in each server node of a cluster by remote login using ssh. The input files describe GA parameters computed by PLGA (section 2.2).

Second, the slave program runs multiple processes of SWAP-DLGA. The processes of SWAP-DLGA in one cluster read the same input file, and then they run with the same GA parameters. A process of SWAP-DLGA is parallelized using the population distribution method (section 2.3) with MPI programming framework. The proposed method runs multiple MPI process of SWAP-DLGA with the same GA parameters. Due to probabilistic behavior in GA, the results of multiple GA processes with the same GA parameter might be different. Thus, the proposed method runs multiple SWAP-DLGA processes with the same GA parameters and computes the average of the results. When all MPI processes of SWAP-DLGA in the cluster finish the computation, the slave program sends output files of SWAP-DLGAs to the master node and ends the process. The output files contain the result of SWAP-DLGA, such as the fitness of their population, the gene values of all individuals, etc. When the master node receives all output files, the node computes the average fitness of each $DLGA_x$ ($F_{avg,DLGA_x}$) and decides whether to run $DLGA_i$ or to delete $DLGA_i$ and to generate $DLGA_{i+3}$. **Table 3** presents the decision strategy of $DLGA_x$ selection.

3.3 Load balance

We implemented the proposed method with the preliminary load balancing strategy in an application level, where we assume homogeneous clusters and they are occupied for the application. The idea of the strategy is to select generation size of each SWAP-DLGA process so that the amounts of computation assigned to clusters are equally balanced.

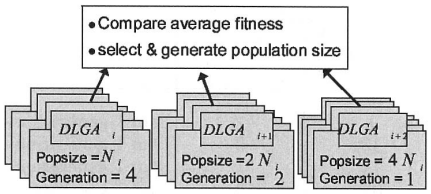

Fig. 4 An example of SWAP-DLGA with PLGA

The computation time of a SWAP-DLGA process ($DLGA_x$), $T_{DLGA,x}$ is defined by eq(4).

$$T_{DLGA,x} = \frac{SWAP(t) \cdot ind(x)}{P} + inGA(t) \cdot gen(x) \quad (4)$$

Here, $ind(x)$ indicates the number of individuals in $DLGA_x$ and is computed by the number of pixels $\times$ the population size. $SWAP(t)$ denotes computation time of the SWAP process, or the evaluation of SimETa in one individual. In our experimental setting (presented in the next section), $SWAP(t)$ distributed from 0.2[sec] through 1.5[sec]. P denotes parallelism. The computation time of the inside GA, $inGA(t)$ is constant because we apply PLGA only for the outside GA. Our preliminary experiments show that $inGA(t)$ is about 40[sec]. The parameter, $gen(x)$, is the number of generation to run $DLGA_x$. $SWAP(t) \cdot ind(x)$ is SWAP computation time in one generation and usually this time is much larger than $inGA(t)$.

The generation size, $gen(x)$, is selected so that $gen(x)$ $\times$ the population size are equal for all $DLGA_x$. For instance, to equalize computation times of three $DLGA_x$s, we set $gen(i)=m^2$, $gen(i+1)=m$, $gen(i+2)=1$, $m=2$, respectively. When all $DLGA_x$ processes finish, the master node gathers all results and compares with their average fitness ($F_{avg,DLGA_x}$). If $F_{avg,DLGA_i}$ is smaller than $F_{avg,DLGA_{i+1}}$, $DLGA_x$ is deleted and the $DLGA_{i+3}$ is generated. Then $DLGA_{i+1}$ is run 4 generations and $DLGA_{i+2}$ is run 2 generations and $DLGA_{i+3}$ is run 1 generation. **Fig. 4** shows an example of SWAP-DLGA with PLGA. In this figure, Popsize denotes the population size of the outside GA and Generation indicates the number of generation to run in the outside GA.

Table 4 The specifications of clusters

Machine	Specification
keio	Xeon E5410 2.33GHz ×2, 11nodes
hiro	Xeon E5410 2.33GHz ×2, 11nodes
kyushu	Xeon E5410 2.33GHz ×2, 10nodes

Table 5 Configration of SWAP-DLGA with PLGA

the outside GA: initial population size	$N_0 = 25$
the outside GA: Serection pressure	$s = 4$
the outside GA: Probability of crossover	$P_c = 0.5$
the outside GA: Probability of mutation	-
the inside GA: population size	1000
the inside GA: generations	1000
the inside GA: Probability of crossover	0.8
the inside GA: Probability of mutation	0.05

Table 6 Configration of original SWAP-DLGA

the outside GA: population size	25
the outside GA: generations	50
the outside GA: Probability of crossover	0.8
the outside GA: Probability of mutation	0.05
the inside GA: population size	1000
the inside GA: generations	1000
the inside GA: Probability of crossover	0.8
the inside GA: Probability of mutation	0.05

4. Experiments

This section presents the experimental results of SWAP-DLGA with PLGA, and compares the performance with original SWAP-DLGA.

4.1 Experimental setting

We conducted the experiments on the distributed computing platform, Intrigger[6]. **Table 4** shows computing resources used in the experiment, where 50 CPU cores in each cluster were used to run the proposed method. Inside a cluster, we run five MPI processes of SWAP-DLGA with the same GA parameters, where each process runs with 10 CPU cores (P=10). **Table 5** and **Table 6** present the configurations of SWAP-DLGA with PLGA and original SWAP-DLGA. PLGA selects suitable population size for the outside GA. We use the satellite images of Suphan Buri Districts's irrigated area in Thailand, observed in 2002, as the input of the experiments. They consist of LR images with ETa of 14 days and HR images with ETa of 2 days.

4.2 Assimilation accuracy

First, we present experimental results to see accuracy of the proposed method. **Fig. 5** presents comparison between ETa and SimETa of LR. The gap between SatETa and SimETa presents the accuracy of the assimilation by SWAP-DLGA, and a smaller gap means better accuracy. From the results in Fig. 5, the assimilation accuracy of LR with PLGA is better than the one with original SWAP-DLGA.

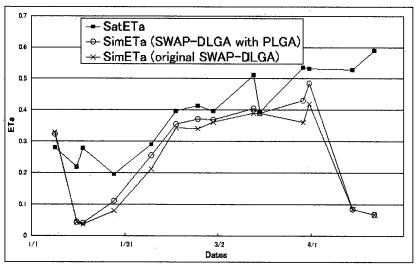

Fig. 5 Assimilation accuracy in LR images

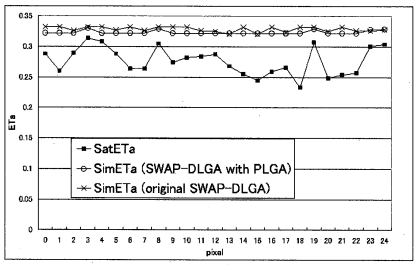

Fig. 6 Assimilation accuracy in HR images (2002/01/08)

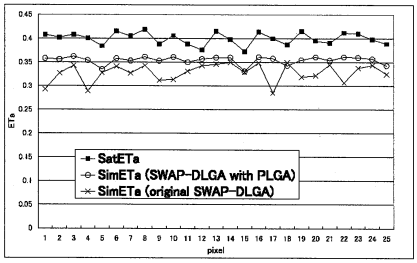

Fig. 7 Assimilation accuracy in HR images (2002/02/16)

Fig. 6, **Fig. 7** presents assimilation results of HR in two different days. In Fig. 7, PLGA produced better assimilation results than original SWAP-DLGA. The results show that the PLGA compute suitable GA parameters and it contributes to improve the assimilation accuracy in SWAP-DLGA.

Fig. 8 presents the outcomes of crop parameters from SWAP-DLGA with PLGA. For the optimized parameters values of GWJan and GWDec are about 56-86cm. The optimized DEC value is within range 4-10 DOY. STS values is within range 28-51 DOY and TEC value is between 95-149 DOY. According to agricultural community, these values are acceptable in rice cropping areas[4].

4.3 The process of increasing fitness

Fig. 9 presents the process of increasing fitness in this experiment. When the generation of

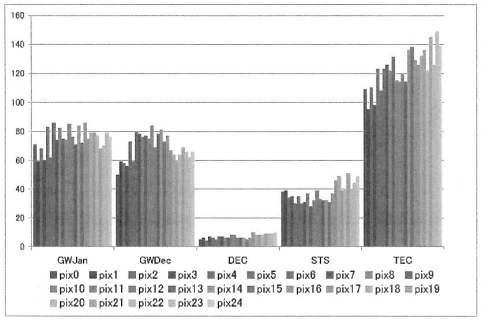

Fig. 8 Values of optimized crop parameters

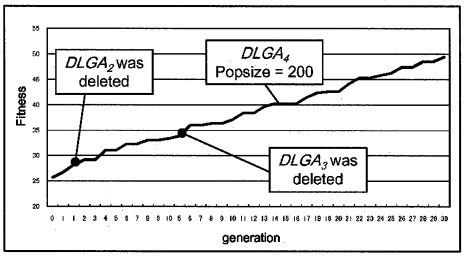

Fig. 9 The process of increasing fitness

$DLGA_3$ was 10, its average fitness $F_{avg,DLGA_3}$ was overtaken $F_{avg,DLGA_4}$. Then, $DLGA_3$ was delieted and $DLGA_4$ was continuously run. When the generation of $DLGA_4$ reached 30, the program was stopped. $F_{avg,DLGA_4}$ was never overtaken the other average fitness. Therefore, we can say that population size = 200 is suitable value as long as within the execution time of this experiment. We didn't know the population size 200 is suitable until we run SWAP-DLGA with PLGA. This highlights the priority to use PLGA on SWAP-DLGA.

4.4 The performance efficiency

Finally, we show the effect of parallelization of SWAP-DLGA on Intrigger. The experimental results show that the computation time of parallelized SWAP-DLGA with PLGA on 150 CPU core, T_D, is 8,749[sec]. Our theoretical analysis shows the approximate computation time on one CPU core, T_S, is 900,000[sec]. So the performance efficiency in computation time (E) is derived from eq(5), where C_t denotes the number of CPU cores.

$$E = \frac{T_S}{T_D} \cdot \frac{1}{C_t} \approx 69[\%] \quad (5)$$

In spite of implementation of the load balancing strategy, the computation times of three $DLGA_x$s are not completely same because our extended experiments show the computation time of one SWAP fluctuates. Improvement of the load balancing mechanism is our future work.

5. Conclusions

This paper proposed SWAP-DLGA with PLGA and presented its implementation scheme on a distributed computing system. The experimental results show that PLGA works well to find suitable population size for the outside GA. Real time monitoring of agricultural activities requires the completion of the computation in 4-5 hours. The execution time of this experiment fulfills the requirement. The proposed method enables to get better crop activity parameters without concerning the setting of the suitable parameters within acceptable time.

The current implementation for load balancing assumes homogenous and dedicate clusters. However, the current distributed computing environment, such as the grid, consists of heterogeneous resources and application performance is affected by external processes competing for the resources. Improvement of the load balancing mechanism is our future work.

Acknowledgment

The work presented in this paper is partially supported by the MEXT Grant-in-Aid for Scientific Research on Priority Area, "Design and Development of Advanced IT Research Platform for Information Explosion Era".

References

1) `http://www.swap.alterra.nl`
2) A. V. M. Ines and P. Droogers, "Inverse modeling to quantify irrigation system characteristics and operational management," *Irrig. Drain. Syst.*, vol.16, no.3, pp.233-252, Aug. (2002).
3) S. Akhter, K. Osawa, M. Nishimura, K. Aida, "Experimental Study of Distributed SWAP-GA Models on the Grid," *IPSJ Trans*, Vol.1 No.2 193-206 (2008)
4) Y. Chemin, and K. Honda, "Spatio-temporal Fusion of Rice Actual Evapotranspiration with Genetic Algorithms and an Agro-hydrological Model," *IEEE Trans on Geogscience and RS*,Vol.44, No.11, 2006
5) G. Harik, and F. Lobo, "A parameter-less genetic algorithm," *GEC conf-99* , July 13-22, Florida(1999)
6) `http://www.intrigger.jp/wiki/index.php/InTrigger`