An Adam-Eve-like Genetic Algorithm and its Application to a Simple Flexible Assembly System

S. A. Feyzbakhsh and M. Matsui

Department of Communications and Systems Engineering, The University of Electro-Communications

Abstract A hybrid Adam-Eve-like GA was introduced and proposed in order to design a simple Flexible Assembly System (FAS). The FAS which is proposed in this article includes a set of removal item case (RIC) of Generalized CSPS workstations. Taking a hint from nature, the proposed method has some special characteristics which distinguish it from others. A comparison between the Adam-Eve-like GA and traditional GAs was conducted. Based on the results from an optimization standpoint, the hybrid Adam-Eve-like GA seems to produce reasonable engineering results in an inherently difficult area where few other techniques are available. Findings suggest that it may be beneficial to apply the Adam-Eve-like GA to other difficult optimization problems.

1 Introduction

We proposed a new implementation of GAs called Adam-Eve GA which has a new genetic operator. Moreover, our GA has become a hybrid one by introducing a reforming feedback path. It has some characteristics which make it a special one. The initial population consisted of only two individuals which made us choose the name of the Adam-Eve-like GA (henceforth referred to as Adam-eve GA). Population size increases during the evolution of population. In other words, new offsprings produced by the crossover operator do not occupy their parent's place but they are inserted as a new individual into the population. Another important point in our proposed GA is consideration of the "lifetime" concept. The new GA operator, " Death" causes this effect. Each individual is exposed to death after a few generations, with a determined probability, which obviously is high.

Taking a hint from the Noah's Ark story, the second implementation of Adam-Eve GA is conducted by, again, just two selected individuals who were found to be the highest fitness during the first implementation. This is just like the animals selected by Noah in his ship who were the initial population of the evaluation of these animals offsprings.

The basic idea of GAs has been taken from nature, therefore it is reasonable if we expect an improvement in our new model for GA, because it is much closer to the evolution of organisms.

The design of an assembly system is a complex task, since there are numerous system parameters and variables involved. We applied our Adam-Eve GA in order to design such kind of a system. In this study, the design process involves approaching a simple FAS as a system in order to determine the required or optimal system parameters and variables under the available technology constraints, and with the objective that the system in question should operate at its desired cost. Therefore, the study focuses on system parameters/variables such as *cycle time, space buffers (storage space in each WS)* and *time buffers (look-ahead times for each WS)*. In view of these, the design optimization problem will be addressed as follows: Given an objective cost function, design the simple FAS to meet (as closely as possible) these given specifications.

2 Methodology of Adam-Eve-like GA

The Adam-Eve GA is a genetic algorithm method and has some differences to the traditional GAs. Maybe a good way to introduce our GA and its differences with traditional GA is to start from the initial population. One of characteristics of our GA which is the reason of naming the method as Adam-Eve GA is to start the population with a very small population such as two individuals where are the parents of whole of successor generations. Adam-Eve GA tries to be more sensitive about the "time". Consideration of the "time" concept can be supposed one of the significant features of the proposed GA, due to its great effect on the evolution. Regarding the time, Adam-Eve GA has an increasing population size during the passage of time which is explained in details later. Furthermore owing to concept of "lifetime" in nature, concept of "death" is added.

Our GA's crossover has an obvious difference with the traditional one. Taking a hint from nature, it produces a new couple of individuals while keeps the parents alive and they can live with their offsprings. It would seem a better model of what happens in nature. Some species which they are short-lived such as some insects , parents lay eggs, and die before their offspring hatch. But in longer-lived species, including human, offsprings live with their parents in the same population. This allows parents to nurture and teach their offsprings but also gives rise to competition between them.

Due to this change in crossover, the population gets larger after each generation. Therefore, it is straightforward that we confront with *Increasing Population Size.*

Another aspect which makes the Adam-Eve GA distinguished from other GAs is the new operator *Death* . In our GA each individual after a determined number of iterations will die out. Such an operator removes (maybe "take the soul" is more meaningful) the individuals from the population after the determined period of lifetime, irrespective of their fitness. It is obvious that also in traditional GAs the low fitness individuals are removed from the population with a high probability and the high fitness ones are kept in the population for an infinite time with a high probability.

Taking a hint from *Noah's Ark Story* we start our GA with *Selected* individuals as initial population instead of random population. Noah built a large ship to save his family and two of every kind of animals from the flood that covered the world. According to this story evolution started by two selected and almost high fitness individuals which he gathered and put in the Ark. In this way we made Adam-Eve GA a hybrid GA since it applies the best results found in one execution of Adam-eve GA as the initial population for the second run.

We restrict the scope of mutation to the new individuals (offspring) produced recently by the crossover operator. As far as our knowledge this is another new feature of our implementation of GA.

As Michalewicz mentioned in [10], it is relatively easy to keep track of the best individual in the evolution process. It is customary (in genetic algorithm implementations) to store "**the best ever**" individual at separate location; in that way, the algorithm would report the best value found during the whole process (as opposed to the best value in the final population).

In the following, the performance of the Adam-Eve genetic algorithm is discussed, briefly. Two individuals are set randomly in a bit fashion, as the initial population. Next step is the evaluating of each generation, using the objective function on the decoded sequences of variables. Then, a same size temporary population are *selected* with respect to the probability distribution based on fitness values. The first recombination operator, i.e., *crossover*, was applied to the individuals in the temporary population. Crossover which is the main genetic operator, operates on two individuals at a time and generates offspring by combining both individual's features. The probability of crossover, P_c, is the ratio of the number of offspring produced in each generation and causes an increase in the the number of population size. This ratio controls the expected number $P_c \times \mathit{CurrentPopSize}$ of individuals to undergo the crossover operation. In this way, for each individual in the temporary population, a random (float) number r from the range $[0 \cdots 1]$ are generated. If $r < P_c$, given individual will be selected for crossover. Then, the selected individuals are mated randomly. Moreover, for each pair of coupled, a random integer number from the range of $[1 \cdots (\mathit{ChromosomeLenght} - 1)]$ is generated, as the position of *crossing point* (*CrossPoint*). Two individuals are generated just like traditional GAs. Although, unlike traditional GAs they are not replaced their parents. They go to the next operation step (mutation) to complete and then make the new individuals who are added to the last population.

As mentioned earlier, *mutation* is the next recombination operator which is performed on a bit-by-bit basis. Mutation is a background operator which produces spontaneous random changes in various chromosomes. The probability of mutation, P_m, is another parameter of the GA which is defined as the percentage of the total number of bits to undergo the mutation operation. $p_m \times \mathit{ChromosomeLength} \times \mathit{CrossNo}$ gives us the expected number of mutated, where *CrossNo* is number of individuals mated by crossover operator in last step. Every bit (in all individuals which were generated by crossover) has an equal chance to undergo mutation (i.e., change from 0 to 1 or vice versa , just like traditional GAs). So, $\mathit{ChromosomeLength} \times \mathit{CrossNo}$ random (float) r from the range $[0 \cdots 1]$ are generated for each bit of

```
begin
  repeat
    $t \longleftarrow 0$
    if ($InitialStage$)
      then random initialize two individuals; $POP(0)$
    else
      then set two "best even"s as initial population; $POP(0)$
    evaluate $POP(t)$
    repeat
        $t \longleftarrow t+1$
        select $POP(t)$ from $POP(t-1)$; $SELECTPOP(t)$
        crossover in $SELECTPOP(t)$; $CROSSPOP(t)$
        mutation in $CROSSPOP(t)$
        evaluate $CROSSPOP(t)$
        increase the population; $POP(t) \longleftarrow POP(t) + CROSSPOP(t)$
        if ($t \geq LifeTime$)
          then die out $POP(old)$ where $old \leq (t - LifeTime)$
    until (termination-condition)[which have not to be the same for initial stage and final stage]
  until (complete of both stages)
end
```

Figure 1: The structure of the Adam-Eve like GA

new offsprings, successively. Moreover, if $r < P_m$ the current bit are mutated.

Then, the new generated individuals by these past operators are added to the last population. Clearly, the population size increase from $CurrentPopSize$ to $CurrentPopSize + CrossNo$.

Now we are ready to apply the next Adam-Eve GA's operator which is *"death"*. Suppose *"LifeTime"* be the number of generations which is assumed the life expectation of each individual. Therefore, each individual after this period of time will die out by a high probability, called P_d.

After *"LifeTime"* iterations the *"death"* operator will be triggered for each individual. The name of each individual with lifetime longer or equal to the *"LifeTime"* is written in the *"Death-list"*. A random (float) number r from the range $[0 \cdots 1]$ is generated for each member of this list. If $r < P_d$, the individuals from current population are removed and the number of population size are updated.

The new individuals has to be evaluated, using fitness function. Now, the whole population and their evaluations are ready to be used to build the probability distribution for the next selection process.

Following selection, crossover, mutation and death complete an iteration of Adam-Eve GA. The rest of the evolution is just cyclic repetition of the above steps. The structure of the Adam-Eve GA is shown in figure 1.

Several possible practical stopping criteria which are used for traditional GAs can also be performed for the Adam-Eve GA.

3 A Simple Flexible Assembly System

The simple FASs consist of a set of WSs that are arranged in a predetermined order and transfer mechanism to convey the usables through the WSs. The models of an assembly system we considered here is a no information feedback simple FAS. Usables are delivered (probably from a central dispatch area) to m functionally identical work stations (WSs) by a conveyor. The topology of the simple FAS under consideration is shown in figure 2. The stations are arranged along the path of the conveyor in the order $1, 2, \cdots, m$. (see figure 2). The WSs may or may not [1] have storage space to store usables received for processing. Usables that cannot be withdrawn from the conveyor by station i will move to station $i+1$ (i.e., the usables is said to *overflow* from station i to station $i+1$), $i = 1, \cdots, m-1$; if a usable on the

[1]For our WSs which are made by Generalized CSPS, if capacity of storage space set to one it performs just like WSs without any storage space.

conveyor is not withdrawn by any of the m stations it will be considered an overflow from the system and stored in an overflow storage area.

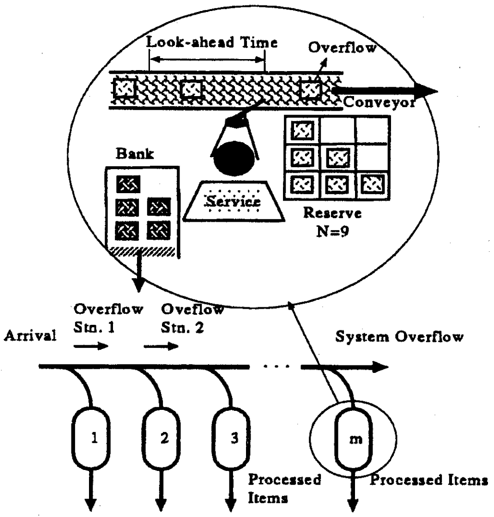

Figure 2: A simple Flexible Assembly System by RIC of Generalized CSPS

A summary of various notations used in this paper are given bellow:

$\bar{x}$	:	mean service time
T_0	:	production planning period
CE	:	constant establish expense for each WS
$\sim_i$	:	$\sim$ for ith WS where i=1,2,...,M
N_i	:	capacity of the reserve
c_{ni}	:	look-ahead time ($n = 1, 2, .., N$)
L_i	:	mean number of in-process inventory
D_i	:	mean delay per service
η_i	:	mean number of overflow per service
α_i	:	unit cost of the number of in-process inventory for T_0
β_{1i}	:	unit cost of each delay's unit time
β_{2i}	:	unit cost of each overflow

Via a simulation, the system-total-cost of the FAS is estimated. The simulation program is written in C++ and is capable of simulating assembly machines (i.e., number of WSs) with a wide range from one to over two hundred assembly stations. In our experiments, each station is assumed to have a constant average of assembly service time which can be different from others in order to cover different service speeds of operators or robots. Each station in the line was assumed to be an unload station, where the usables were unloaded from the conveyor and the remaining usables (overflow) were transfered to the next station upon arrival. We assumed that the system arrival time and each station's service time were random variables with Erlangian distributions. These are referred to as the delay-idle time of the operator and the overflow from the last station.

Also, the total variable cost associated with a WS, as calculated over some convenient time period, as the objective function for the multi-station of RIC of Generalized CSPS (using the single station production cost introduced in [7] [8] [9] [?] [?] [3]) is as follows:

System-Total-Cost=

$$m \cdot CE + \sum_{i=1}^{m} \alpha_i \cdot L_i + \left(T_0 \big/ (\bar{x}_i + D_i)\right) \cdot \left((\beta_{1i} \cdot D_i) + (\beta_{2i} \cdot \eta_i)\right) \tag{1}$$

4 Numerical Considerations

This section investigates the ability of the proposed method in comparison to the traditional GAs. Here, we will stress the solution to the design optimization problem, described before, through different examples. In order to solve the problem, we will then input the calculated parameters (i.e, required number of WSs and the cycle time) to the simulation model of the simple FAS. Integrating the simulation model in the Adam-Eve GA and the traditional GA, we will optimize the buffer sizes of our FAS in order to reduce the delay and overflow effects as much as possible. Under some assumptions and given fixed parameters for a 10-station system, the cycle time and arrive rate are obtained 0.1264 and 7.9, respectively. Furthermore, we integrate the discrete event simulation model of the simple FAS into a genetic algorithm (GA) and the Adam-Eve GA procedures to optimize the buffer sizes. We are interested in whether the Adam-Eve GA can outperform a traditional GA.

The configuration was tested allowing all buffer storage capacities (i.e., capacities of the reserves for each station) to vary from 1 to 10 units. Moreover, the first look-ahead times (i.e., c_1, when the reserve is almost full) vary from 0.1 to 1.6. These variables constraints for c_2 are 1 and 2.5, and also, for c_3 are 2 and 5.1. The step size of these variables are 0.1. It was then possible to present individuals station capacity of reserve in four-bit, also each of c_1 and c_2 in four-bit, and c_3 in five-bit position allowing the system with ten stations to be defined by a string of length 170, clearly. By an expensive experiment the best good set of traditional GA was found. Therefore, the traditional GA runs were performed having population sizes equal 30 and $P_c = 0.6, P_m = 0.001$. The first and second tables show the results of the Adam-Eve GA program and a traditional GA, respectively. Each table presents 5 different runs of the optimization methods which are sorted according to the quality of the solutions. The tables include the optimum variables found for the first and the 7th stations. The two last columns of each table indicate the system-total-cost and the elapsed time that the corresponding GA takes to evolve the population, respectively.

As we can see from the tables, the Adam-Eve GA found the optimum solution approximately fourteen times faster than the traditional GA. Regarding the variables found by each method, we can compare their convergence. As can be seen, Adam-Eve GA converges more than the traditional one.

Regarding computational time, it may concluded that the Adam-Eve GA outperforms traditional GAs. Moreover, fairly accurate results in the examples were conducted by Adam-Eve GA support this argument.It may be due to the fact that Adam-Eve GA is likely a better model of what happened in nature.

Table 1

Adam-Eve GA for 10 stations

RUN	Station 1 Capacity	Station 1 Look Ahead 1	Station 1 Look Ahead 2	Station 1 Look Ahead 3	Station 7 Capacity	Station 7 Look Ahead 1	Station 7 Look Ahead 2	Station 7 Look Ahead 3	Total Cost	Elapsed Time
1	1	0.57	3.00	3.87	1	0.57	2.68	3.23	14681	00:17:46
2	1	0.57	2.48	3.87	1	0.57	2.68	3.23	15333	00:15:57
3	1	0.57	3.00	3.87	1	0.57	2.68	3.16	15342	00:16:10
4	1	0.57	2.35	3.87	1	0.57	2.68	3.23	15366	00:15:47
5	1	0.57	3.00	3.87	1	0.57	2.68	3.23	16212	00:11:51

Table 2

Traditional GA for 10 stations

RUN	Station 1 Capacity	Station 1 Look Ahead 1	Station 1 Look Ahead 2	Station 1 Look Ahead 3	Station 7 Capacity	Station 7 Look Ahead 1	Station 7 Look Ahead 2	Station 7 Look Ahead 3	Total Cost	Elapsed Time
1	2	0.57	1.52	3.81	1	1.22	1.07	2.97	19260	03:54:45
2	1	0.57	1.07	3.48	1	0.38	2.81	2.06	20447	04:54:43
3	1	0.57	3.00	3.87	1	0.57	2.68	3.23	20962	04:04:23
4	5	0.57	1.77	3.10	3	0.29	1.71	2.90	22431	03:54:42
5	3	1.22	2.03	3.80	3	0.85	1.84	2.19	22947	03:54:40

5 Conclusions and Extensions

We have introduced and proposed a hybrid Adam-Eve GA with application to the simple FAS design optimization problem under study. The proposed method has been compared to traditional GAs. Based on the results from an optimization standpoint, the hybrid Adam-Eve GA seems to produce reasonable engineering results in an inherently difficult area where few other techniques are available.

The computational requirements for a traditional GA run were quite large. Moreover, in the examples provided in the previous section, although there are no significant differences between the results (i.e., estimates of minimum total-system-cost) obtained by these two algorithms, we can find that, in comparison to a traditional GA, the computation time decreased significantly by using the Adam-Eve GA.

A time of 70 minutes (on a Pentium-200 PC) was required to complete 100 iterations, for the 3-station simple FAS, of a traditional GAs. This computation time increased to around 8 hours, for 10-station, under the same conditions. Typical execution times for the Adam-Eve GA implementation were approximately less than 11 and 30 minutes for 3-station and 10-station respectively.

Applying this method, we will be able to narrow the range of values for some of the decision variables, for example, the capacity of the reserve and various look-ahead times for each WS, as well as the number of WS and arrive rate.

Adam-Eve GA can also be an effective tool to study the system-total-cost and evaluate the effects of different variables on in-process inventories, as well as delay and overflow of each station.

However, our experiments were limited to a simple FASs, and so it may be beneficial to apply the Adam-Eve GA to other difficult optimization problems.

Future research areas to be investigated include:

- application of the Adam-Eve GA to some other difficult optimization areas.
- The comparison of the Adam-Eve GA with some other optimization methods.

References

[1] Beasley, D., Bull, D.R., and Martin, R.R., An overview of Genetic Algorithms: part 1, Fundamentals, University of Cardiff, University Computing, 1993, 15(2) 58-69.

[2] Buzacott, J.a. and Shantikumar, J.G., "Stochastic Model of Manufacturing Systems", Prentice-Hall, Inc (1993)

[3] Feyzbakhsh S.A., *Optimal Design of a Generalized Conveyor-Serviced Production Station: Fixed and Removal Item Cases*, Proceedings of the 6th IFIP TC5/WG5.7 International Conference on Advances in Production Management Systems - APMS'96, Kyoto Japan, 1996, 209-214.

[4] Gen, M. and Cheng, R., *Genetic Algorithm & Engineering Design*, John Wiley & Sons, 1997.

[5] Goldberg D.E., *Genetic Algorithms in search, Optimization, and machine Learning*, Addison-Wesley, New York, NY (1989).

[6] Holland, J.H., *Adaptation in Natural and Artificial Systems*, MIT Press, 1975.

[7] Matsui M., "A Generalized Model of Conveyor-Serviced Production Station (CSPS)", *Journal of Japan Industrial Management Association*, Vol. 44, No.1(1993) 25-32 (in Japanese).

[8] Matsui M., Shingu T. and Makabe H., "An Analysis of Conveyor-Serviced Production Station by Queueing Theory", *Journal of Japan Industrial Management Association*, Vol.28, No.4 (1978) 375-386 (in Japanese).

[9] Matsui M., Shingu T. and Makabe H., "A Comparative Consideration of Operating Policies for Conveyor-Serviced Production Station", *Journal of Japan Industrial Management Association*, Vol.31, No.3 (1980) 342-348 (in Japanese).

[10] Michalewicz Z.,*Genetic Algorithms + Data Structures = Evolution programs 3rd rev.*, Springer-Verlag,1996.