The final version of this paper appears in *Computers and Chemical Engineering, Vol. 57, pp. 112–121 (2013).*

# Generation of operating procedures for a mixing tank with a micro genetic algorithm

Rafael Batres

Industrial Systems Engineering Group

Department of Mechanical Engineering

Toyohashi University of Technology

Hibarigaoka 1-1, Tempaku-cho

Toyohashi 441-8580, Japan

E-mail: rbp@tut.jp

**Abstract**. This paper explores the use of a micro genetic algorithm that uses variable-length chromosomes and a seeding scheme based on tabu search. The problem is to find the sequence of actions that have to be executed in the shortest time possible, but also in a way that minimizes the possibility of situations that may endanger the plant personnel and plant facilities. The proposed approach was tested on the generation of the optimum sequences for startup and shutdown of a mixing vessel similar to the equipment used in the synthesis of acrylic acid. The results show that the proposed method outperforms the traditional GA algorithm both in terms of the quality of the solution and computational effort.

Keywords: operating procedures, micro genetic algorithms, tabu search, mixing tank

## 1. Introduction

A significant number of serious incidents occur in the process industry during shutdown or startups operations (IChemE, 2006). However, contrasting to industrial applications of scheduling algorithms, the generation of operating procedures is still left to expert engineers and plant personnel.

In an early study, Benson and Perkins (1997) concluded that improving the control of startups, shutdowns and grade changes presented a potential of world benefits of about 100 million US$ per year. More recently, a site study conducted by the Abnormal Situation Management Consortium revealed the average cost of $2.6 Million per year due to incidents that had procedural operations as a contributing factor (Kucharyson, 2006).

The generation of operating procedures can be described as a planning problem where the objective is to find an ordered sequence of plant actions to take the process from an initial state to a goal state. In general, the sequence of actions needs not only to be carried out in the shortest time possible, but also in a way that minimizes the possibility of situations that may endanger the plant personnel or cause damages to the plant.

The generation of operating procedures has a relatively long history going back to the work of Rivas, Rudd and Kelly (1974) whose method is based on the General Problem Solver (Batres, Soutter, Asprey, & Chung, 2002). The majority of the approaches for the generation of operating procedures falls into one of two categories: state-based planning or simulation-based planning approaches.

State-based planning approaches have origins in the artificial intelligence planning techniques. These planning methods represent operations as a function that consists of preconditions and effects represented as first-order predicate calculus propositions. Such representation approach

works well for static environments, but is inappropriate for many real problems where the duration of an action should be considered.

One example of a state-planning approach is the method proposed by Fusillo and Powers (1987) who implement a planner in which the problem is stated as a state-space graph where operations are used to move between states. In their work, a state is defined as a vector that combines qualitative values of process variables with positions of valve actuators and pump switches.

A specific kind of state-planning approach is given by the action ordering systems in which the actions are given by the design of the plant. For example, Lakshmanan and Stephanopoulos (1988) introduced a methodology based on the planning approach proposed by Chapman (1987). Their methodology incorporates the use of a hierarchical representation of the plant structure, which is utilized for constraint propagation and subgoaling, as a strategy to reduce the search space. Another strategy in their methodology is the use of temporal constraints and binary qualitative mixing constraints. Temporal constraints are expressed in terms of precedence relations such as *T* should precede *S*. An example of their mixing constraints is "*A* and *B* should not come into contact with each other."

Soutter and Chung (1995; 1996) describe the Chemical Engineering Planner (CEP), which implements a non-linear, partial-order algorithm similar to the approach used by Weld (1994). In CEP, goals are translated into subgoals by means of an inference engine that uses backward chaining. This planner also implements domain knowledge for the representation of the plant topology and constraints. Specifically, constraints represent situations to be avoided such as those that prevent a heat-exchanger to be operated before starting a pump that creates a flow through the exchanger.

Some attempts have also been made to generate operating procedures for batch processes.

Viswanathan *et al*. (1998; 1998) describe an interactive technique that uses a hierarchical description of operations based on the procedural representation defined in ISA S88 (ISA, 1995). The construction of the procedures starts with the generation of a graph in which some of the nodes represent material flows and others represent unit procedures. Subsequently, depth-first search is used to find paths that connect raw materials with products. Finally, equipment is assigned to the unit procedures by matching against the equipment capabilities.

While some researchers identified the necessity of providing optimum plans, the above mentioned methods focused on feasibility rather on optimization of time or cost. Another limitation of such planning methods is that they can only handle qualitative constraints such as 'chemicals *A* and *B* should not be mixed together.' Simulation-based approaches attempt to address both issues.

However, little has been reported on efforts for determining operation sequences in the presence of quantitative safety constraints and dynamic behaviors. In this vein, Yang, Li, and Xu (2011) emphasize the benefits of dynamic simulation for identifying optimal startup procedures. For example, improvements can be made to operate a unit to obtain safer startup trajectories that reach the goal in less time.

Mixed-integer dynamic optimization (MIDO) that combines dynamic optimization with discrete variables can be used to obtain optimal sequences such as in the $N_2$, $O_2$, $CH_4$ mixture changeover reported by Galán & Barton (1997) and Barton, Banga & Galán (2000). However, such approach is useful in situations where a globally optimal solution is not required and one can settle for plan feasibility.

In another effort, Asprey *et al*. (1999) proposed a two-layer method to generate startup sequences for a mixing problem similar to the one described by Galán and Barton (1997). The method proposed by Asprey *et al*. explicitly takes into account time optimality while maintaining

realizable actions (e.g. their approach prevents actions such as opening a valve 0.05% for 0.3 seconds, which is not physically realizable). In order to guarantee realizable actions the time dimension is discretized through the introduction of the *operator time constant*, which denotes the time duration for which a combination of valve positions is maintained. The upper layer uses Simulated Annealing to optimize the overall operations time by adjusting the *operator time constant*, which is passed to the lower layer. In the lower layer, a path-constrained optimization determines the sequence of valve operations that minimize the difference between the current mixture composition and the goal state. Due to the fact that the quantitative constraint (a flammability envelope) introduces non-convexity into the problem, the A* (read A-star) search method is introduced. Despite the fact that the two-layer method has the ability to generate a global optimal solution, it does so at the expense of a high number of function evaluations. Moreover, keeping the same operator-time-constant for all the operations misses solutions that would otherwise be obtained with heterogeneous values for this parameter.

As investigated by Asprey *et al*., path constraints such as those associated to flammability envelopes result in non-convex optimization problems. For this reason, in this paper we investigate an approach based on micro genetic algorithms ($\mu$GAs) and variable-length chromosomes to generate startup and shutdown operating procedures.

$\mu$GAs are characterized by a small population size and consist of restarting the population several times while keeping the very best fit individual (Krishnakumar, 1989). Thanks to the small populations, convergence can be achieved faster and less memory is required to store the population.

Another feature of the proposed approach is the use of variable-length chromosomes. Perhaps the earliest attempt to use variable-chromosomes is the work of Robbins (1995), who applied a variable-length representation to solve the traveling salesman problem. A simple

alternative is the use of a fixed-length chromosome. However, a fixed-length chromosome would have to be sufficiently large, resulting in some of the genes actually active and the rest inactive. As a result, fixed-length chromosomes would require more memory.

Variable-length chromosomes have been applied with success in artificial intelligence planning in situations that share similar characteristics to the problem considered in this paper. In a planning problem, the objective is to find an ordered sequence of actions that achieve a goal given an initial state, provided that templates for those actions are available. For example, Westerberg and Levine (2001) use variable-length chromosomes to represent feasible but not necessarily optimum plans. Their GA implementation extends traditional algorithms by incorporating a shrinking operator that deletes a randomly selected action from the parent chromosome. Brie and Morignot (2005) describe a genetic algorithm that besides the shrinking operator of Westerberg and Levine also uses an operator for enlarging a chromosome (by inserting new actions), an operator for swapping two genes, and an operator that modifies a parameter of a randomly selected action. In these two variable-length chromosome implementations, the respective algorithms are applied to the so-called blocks-world problem.

This paper is structured as follows. Section 2 describes the problem statement. Next, the proposed methodology is presented in Section 3. Then, Section 4 discusses the numerical experiments that were conducted to evaluate the proposed approach. Finally, Section 5 presents the conclusions and discussion.

## 2. Problem statement

We adopt time optimality as a performance criterion. Therefore, the general problem for generating operating procedures for startup or shutdown can be written as follows:

Min $A\int_{t_0}^{t_f} dt + B\left\|\boldsymbol{x}_f - \boldsymbol{x}(t_f)\right\|$ (1)

subject to $\boldsymbol{f}\left(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{u}(t)\right) = 0$ (2)

$\boldsymbol{x}(t_0) = \boldsymbol{x}_0$ (3)

$\boldsymbol{g}(\boldsymbol{x}) \leq 0$ (4)

$\boldsymbol{u}(t) \in U,\ U = [u_1, u_2, \ldots, u_n]^T$ (5)

where $\int_{t_0}^{t_f} dt$ is the time that the system takes to get from the initial state to the final state; $\boldsymbol{x}(t)$ represents the state that characterizes the evolution of the system through time; $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$ are vectors that represent the initial state and final state respectively; $\boldsymbol{u}$(t) is a vector that represents a set of operations performed at time $t$; $\boldsymbol{f}\left(\boldsymbol{x}(t), \dot{\boldsymbol{x}}(t), \boldsymbol{u}(t)\right)$ are a set of algebraic differential equations describing the process behavior; $\boldsymbol{g}(\boldsymbol{x})$ is a vector of inequalities representing process, safety, and other constraints; $U$ represents a set of predetermined operations such as specific valve positions; and $A$ and $B$ are constants. When $A = 0$ the problem is reduced to finding the sequence of operations that result in a feasible trajectory but not necessarily optimum.

A feasible solution is one that achieves the goal without violating any constraints. The constraints include those imposed by the chemistry of the process, product quality requirements, process constraints (e.g., such as those related to catalyst decomposition), safety constraints (flammability, explosiveness, and toxicity), and mechanical constraints (design temperature and pressure limits) (Batres, Soutter, Asprey, & Chung, 2002).

The system under study is a mixing vessel that belongs to the kind of equipment normally used in the synthesis of acrylic acid that is also described by Asprey *et al.* (1999). The mixing vessel is a stirred-tank with three inlet valves and one outlet valve as shown in Figure 1. There is one inlet valve for the admission of steam (valve v-1), one for propylene (valve v-2), and another

for air (valve v-3). The outlet valve is for the discharge of the mixture. The inlet valves can be operated in a bang-bang fashion or with multiple valve positions. The outlet flow is regulated with a local controller that keeps the pressure constant during both operation modes. All the valves are assumed to behave ideally.

The objective is to determine the optimum sequence of operations that minimize the risk of a flammable mixture (entering the flammability zone poses a fire and explosion hazard).

In the startup operations, the vessel is initially filled with air and the final state is determined by a specified mass composition of the steam-air-propylene mixture outside the flammable envelope. This is the case for placing the vessel back into service.

Conversely, the shutdown operations have the goal of transferring the system to a state that is 100 % air by mass, in a way that does not produce flammable mixtures. This is the case of a vessel-entry procedure (Crawl, 2012).

As mentioned previously, valve operations should not take the process through the flammable envelope. Therefore, a feasible trajectory for the mixing vessel is that in which the final state is reached and the valve operations are carried out to avoid unsafe mixtures. The initial, intermediate, and final states are all expressed in terms of mass fraction. The opening or closing of valves results in a new state, which is calculated using dynamic simulation. In this work, a lumped parameter model is used. In all cases, ideal mixing is assumed.

## 3. Methodology

To solve the mixing problem, we use the micro genetic algorithm ($\mu$GA) shown in Figure 2. The first step is to generate a random population. The next step is to use tabu search to find a good feasible solution and insert it to the population.

The inner loop in Figure 2 is essentially a traditional GA algorithm (albeit with variable-length chromosomes) consisting of the evaluation of the fitness of each member of the population, the selection of parent chromosomes, the generation of a new population by means of the crossover[1] and mutation operations, and the separation of the best-fit individual after convergence. The inner loop implements the roulette-wheel scheme (Davis, 1991) for the selection of the parent chromosomes.[2]

The outer loop consists of creating a new random population, transferring the best individual from the inner loop, and restarting the inner loop. In this paper, each cycle in which the inner loop is restarted is called an *epoch*.

### 3.1 Seeding strategy

Normally, a genetic algorithm starts with an initial population generated at random. However, several studies indicate that seeding GAs with good estimates may result in better solutions, especially when the good estimates are obtained with a low computational effort (Keedwell & Khu, 2005). In order for this approach to be effective, the seeding algorithm must be computationally faster than the GA itself.

In the proposed methodology, the initial population is obtained by incorporating a single good feasible solution to a randomly generated population. This good feasible solution is found by solving a problem that is formulated as an AI planning problem in which the objective function is based on a measure of the distance of the path from the initial state to the current node and a measure of the distance of the path from the current node to the goal. It was found

[1] Besides this basic crossover, two variations are implemented: elitist crossover (Kubota, Neya, & Taniguchi, 2002) and crossover-and-mutate.

[2] Although, in general, tournament selection is more effective, when tested on the mixing tank problem, tournament selection resulted in a few comparatively highly fit (but not optimal) individuals that came to dominate the population causing the convergence to a local optimum.

that the Chebyshev distance performs best among other distance approaches considered. Therefore, the objective function is formulated as:

$$f(\mathbf{y}_n, n) = w(n)A\exp\left(\max_i\left|y_{n,i} - y_i^{goal}\right|\right) + B\max_i\left|y_{n,i} - y_i^{start}\right| + p(y_n, n) \quad (6)$$

where $\boldsymbol{y}_n$ is a state vector containing the mass fractions that result from executing an operation; $n$ is the number of operations carried out so far that is less or equal to the final number of operations that reach the goal state; $w(n)$ is an exponential decay function that reduces its value as the solution progresses towards the goal state; $y_{n,i}$ represents the mass fraction of component $i$; $y_i^{\text{start}}$ represents the mass fraction of component $i$ at the initial state; $y_i^{\text{goal}}$ represents the mass fraction of component $i$ at the goal state; and $p$ is a penalty function that is active when the flammability constraint is violated and whose value decreases as the solution reaches the goal state. $p$ is given by

$$p(\mathbf{y}_n, n) = F\left(1 + q(\boldsymbol{y}_{n-1}, \boldsymbol{y}_n)\right)\left(1 - \frac{n}{N}\right) \quad (7)$$

where $q(\boldsymbol{y}_{n-1}, \boldsymbol{y}_n)$ is the maximum difference between the gas concentration and the flammability envelope for all the concentrations that fall in the flammability envelope as part of the path that results from moving from state $j-1$ to $j$. If moving from state $j-1$ to $j$ results in a path that is outside the flammability envelope on all its segments then $q(\mathbf{y}_{n-1}, \mathbf{y}_n) = 0$.

It is assumed that when gas flows into the tank, the gases are completely mixed so that a uniform composition is obtained. However, in some situations, compositions are not uniform and dangerous flammable-mixture zones may develop. For this purpose, a safety margin of one

standard deviation value is included around the flammable envelope. The extended flammability envelope is approximated by an eighth-degree polynomial. The resulting mathematical expression of this polynomial is:

$$y_{\text{steam}} = -4.854787997 + 589.0562329 y_{\text{prop}} - 28089.2016 y_{\text{prop}}^2 + 731729.9028 y_{\text{prop}}^3 - 11378315.63\, y_{\text{prop}}^4 + 108305115.6\, y_{\text{prop}}^5 - 618545476.2\, y_{\text{prop}}^6 + 1945639394 y_{\text{prop}}^7 + 2.590347960 \times 10^9 y_{\text{prop}}^8 \quad (8)$$

Tabu search is used as the optimization algorithm to obtain the good feasible solution. The specific tabu search algorithm is described in the next subsection.

### 3.1.1 Tabu search algorithm

Tabu search is a stochastic optimization approach that makes use of a short-term memory in the form of a tabu list (Glover, 1989). Tabu search uses a guided local search procedure that avoids local optima and rejects moves to points already visited in the search space.

Tabu search has been shown to be effective for many engineering optimization problems, especially for combinatorial optimization problems. Tabu search first generates a set of trial solutions (neighbors) in the neighborhood of an initial solution. Then the procedure moves to the best trial solution even if no neighbors are better than the initial solution. In this work, the neighborhood is obtained from a random selection of different combinations of valve positions and the operator-time-duration, which is the time in which the input valves remain at a certain position (equivalent to the *operator time constant* introduced by Asprey *et al*.).

In order to avoid visiting the same solutions, the algorithm consults a list known as *tabu list* that contains the last few solutions. In addition, a record of the best solution ever found ($u_{\text{best}}$) is separately maintained. The algorithm with small variants consists of the following steps:

Step 1. Initialize the list of operations $R = \emptyset$ and the list of states $S = \emptyset$.

Step 2. Set depth $n = 0$ then set the initial state as $y_{\mathrm{n=0}}$ and insert it in $S$.

Step 3. Randomly select a single operation $u_0$ from $U$. Simulate the execution of this operation to determine the resulting new state (in terms of mass fraction), and evaluate it using the objective function. Set $u_{\mathrm{best}} = u = u' = u_0$. Initialize the tabu list by setting $T = \emptyset$.

Step 4: Randomly select a set of operations from $U$, simulate each of them to determine the resulting mass fractions, and evaluate each of them using the objective function. These operation candidates become the neighbors of $u$. Then, sort the selected operations based on their objective function values. Select the best candidate from this set and assign it to $u'$. Set the mass fraction that results from executing $u'$ to $y'$.

Step 5. If $u'$ is in the tabu list then check the aspiration criterion of $u'$. If the aspiration criterion is met then go to Step 6, otherwise delete $u'$ and select another neighbor of $u$. If $u'$ is not in the tabu list, then put it in $T$.

Step 6. If the objective function of $u'$ is better than that of $u_{\mathrm{best}}$, set $u_{\mathrm{best}} = u = u'$, else go to Step 7.

Step 7. If the upper limit of iterations since the best solution was updated has been reached then set $u_{\mathrm{best}} = u'$, else set $u = u'$.

Step 8. If $u = u'$ then set $y_{\mathrm{n+1}} = y'$, insert $y_{\mathrm{n+1}}$ in $S$ and $u$ in $R$. Then set $n = n + 1$.

Step 9. Check the termination criteria. If the criteria are satisfied then stop, otherwise go to Step 4.

Step 10. Return the list of operations $R$.

### 3.2 Objective function of the $\mu$GA

The objective function of the inner loop incorporates the total operations time along with a feasibility term, and two positive terms that become active when the flammability constraint is violated. The objective function is given by Equation (9).

$$f(n) = w_1 t_{\text{total}} + w_2 \varphi + av + bq(\boldsymbol{y}_{\text{n}-1}, \boldsymbol{y}_{\text{n}}) \quad (9)$$

where $t_{\text{total}}$ is the total time that takes from the initial state to the last state; $\varphi$ is a function of the distance between the last state and the goal, and the distance between the last state and the initial state; $v$ is a penalty function that is zero if no violation occurs and positive otherwise; $q(\boldsymbol{y}_{\text{n}-1}, \boldsymbol{y}_{\text{n}})$ is the same penalty function described in Section 3.1; and $w_1$ and $w_2$ are weights. The weight $w_1$ is set to a higher value ($w_{\max}$) at generation 0, then it is linearly decreased as the iteration progresses with generation $k$ and it is calculated by Equation (10).

$$w_1 = w_{\max}/(1 + mk) \quad (10)$$

Function $\varphi$ is defined as:

$$\varphi = D\exp\left(\max_i \left| y_{n,i} - y_i^{goal} \right|\right) + E \max_i \left| y_{n,i} - y_i^{start} \right| + L \frac{\sqrt{\sum_i \left(y_i^{goal} - y_{n,i}\right)^2}}{\sqrt{\sum_i \left(y_i^{goal} - y_i^{start}\right)^2}} \quad (11)$$

which combines the Chebyshev distance with a function based on the Euclidian distance. These two distances are combined such that if both $D$ and E take non-zero values, $L$ will be zero, and vice versa.

Penalty function $v$ is defined as:

$$v = \left(\sum_{k=0}^{n} v^k\right)^2 \quad (12)$$

where

$$v^k = \begin{cases} 1 & \text{if the path from state } k-1 \text{ to state } k \text{ enters the envelope} \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

Equation (13) is interpreted as follows. If the flammability constraints is not violated the value of the violation penalty is zero. The violations that result after executing some of the operations are then summed.

The fitness function used by the genetic algorithm is calculated as the inverse of the objective function:

$$fitness = \frac{1}{f(n)} \quad (14)$$

where $f(n)$ is given by equation (9).

### 3.3 The operations chromosome

As in traditional genetic algorithms, each candidate solution in the population is represented by a data structure called chromosome. In the current problem, each gene in the chromosome represents an operation defined in terms of positions of the three valves, and a value for the operator time duration.

Each operation is indexed and stored in an *operation pool* before the actual optimization begins. For example, the chromosome shown in Figure 3 has two genes with index 59 each of which represents an operation in which the steam valve is fully closed, the propylene valve is 10% open, the air valve is fully open, and these positions are kept unchanged during 30 seconds.

The chromosome structure is defined so that genes can be added or removed. In other words, this is a variable-length chromosome.

### 3.4 Mutation operators for the variable-length chromosome

In addition to the two-point crossover and the mutation operator, the shrink, growth, swap, and parameter-change mutation operators (Brie & Morignot, 2005) were implemented.

The shrink mutation consists of picking one random point in the chromosome of the parent and then removing the operation at that specific point.

The growth mutation is carried out by selecting a random point in the chromosome of the parent and then inserting one operation from the operation pool at that point. The operation from the operation pool is also selected at random.

The swap mutation consists of randomly selecting two positions in the chromosome and then swapping their respective genes.

The parameter-change mutation is similar to the traditional mutation, except that the operation from the operation pool has the same valve positions as the mutation target but different time duration.

## 4. Numeric experiments

Experiments were carried where the $\mu$GA was compared to two alternate algorithms: a traditional genetic algorithm with large populations (LPGA) and a genetic algorithm (TSGA) that uses the seeding strategy based on tabu-search that is described in Section 3.1.1. In order to evaluate the repeatability of the results, 10 runs were carried out for each algorithm.

The three algorithms were programmed in Java. The object-oriented features of this computer language facilitated the implementation and reuse of individual objects representing: the process-behavior simulator for the mixing model; the output valve controller; and the valve sequencing methods. The mixing model was solved using a fourth-order Runge-Kutta method

implemented in the Michael Thomas Flanagan's Java Scientific and Numerical Library (Flanagan, 2010). However, the object-oriented implementation makes it possible to replace the process-behavior simulator without having to modify the controller and valve sequencing parts.

The experimental results were obtained on a 3.2 GHz Intel Xeon computer with 8GB of RAM and Windows 7. All random numbers used in the algorithms were generated using the Mersenne Twister random number generator (Matsumoto & Nishimura, 1998).

All the experiments were carried considering a 50 ℓ tank with a pressure of 1 atm and a temperature of 500 K. The maximum flow rates of each inlet stream were set to 0.1 kg/s. The operator-time-duration was set for 15, 21, or 30 seconds. The positions of the inlet valves were considered as completely-open, completely-closed, or 10% open. The output valve was operated using a proportional controller with a gain equal to 3.

Both startup and shutdown cases were investigated. In the startup case, the initial state was 100 % air by mass, and the goal state was 10% steam by mass, 15% of propylene by mass and the balance was air. Other goal states were explored but this one resulted the most challenging in terms of the number of local minima. In the shutdown case, the initial and goal states were selected as the inverse of those of the startup.

The population of the $\mu$GA consisted of 5 individuals. The termination criteria for the inner and outer cycle were set to a maximum of 40 generations and 20 epochs respectively. The values of the parameters used in the experiments are shown in Appendix 2.

Both LPGA and TSGA were configured to accommodate a population of 100 individuals. The termination criterion for LPGA was defined based on a maximum number of generations after exploratory runs showed that convergence stabilized at 250 generations. However, for analysis purposes, convergence was considered if the best-fit individual remains unchanged for 20 successive generations. It was confirmed that the converged value also remain unchanged

until the 250$^{th}$ generation (Figure 8).

Figures 4 and 6 compare the process (state) trajectories for the startup and shutdown cases, respectively. The respective operation profiles of these trajectories are shown in Figures 5 and 7. Tables 1 and 2 summarize the results of the numerical experiments.

The operating procedures can be obtained from the optimum chromosome. For example, the optimum chromosome for the startup case is: {70, 70, 70, 70, 18, 13, 59, 58, 59, 6, 6, 58, 58, 58, 58, 58, 58}. Here, the first five operations with numbers 70 and 18 have the same valve positions. Similarly, the last eight operations with numbers 6 and 58 have the same valve positions. Therefore, contiguous operations that have the same valve positions can be combined by adding their individual operation times.

The resulting operating procedures can be represented using time conditions:

At $t$ = 0 seconds set v-1 to 100%, v-2 to 10%, v-3 to 0%
At $t$ = 135 seconds set v-1 to 10%, v-2 to 100%, v-3 to 10%
At $t$ = 150 seconds set v-1 to 0%, v-2 to 100%, v-3 to 100%
At $t$ = 180 seconds set v-1 to 0%, v-2 to 10%, v-3 to 100%
At $t$ = 210 seconds set v-1 to 0%, v-2 to 100%, v-3 to 100%
At $t$ = 240 seconds set v-1 to 0%, v-2 to 10%, v-3 to 100%

Alternatively, process states can be used instead of time:

When $y_{\text{steam}}$ = 0.0% and $y_{\text{propylene}}$ =0.0% and $y_{\text{air}}$ =100%, set v-1 to 100%, v-2 to 10%, v-3 to 0%
When $y_{\text{steam}}$ = 33.947% and $y_{\text{propylene}}$ =3.4003% and $y_{\text{air}}$ =62.6652%, set v-1 to 10%, v-2 to 100%, v-3 to 10%
When $y_{\text{steam}}$ = 32.4253% and $y_{\text{propylene}}$ =8.1492% and $y_{\text{air}}$ =59.4374%, set v1 to 0%, v-2 to 100%, v-3 to 100%
When $y_{\text{steam}}$ = 26.6368% and $y_{\text{propylene}}$ =15.6202% and $y_{\text{air}}$ =57.7526%, set v-1 to 0%, v-2 to 10%, v-3 to 100%
When $y_{\text{steam}}$ = 23.9827% and $y_{\text{propylene}}$ =14.9696% and $y_{\text{air}}$ =61.0564%, set v-1 to 0%, v-2 to 100%, v-3 to 100%
When $y_{\text{steam}}$ = 19.9235% and $y_{\text{propylene}}$ =20.8987% and $y_{\text{air}}$ =59.185%, set v-1 to 0%, v2 to 10%, v-3 to 100%

Similar to situations common to other planning problems, the objective function played an important role in the quality of the results. In the startup case, setting $\varphi$ to $\varphi = D\exp\left(\max_i \left|y_{n,i} - y_i^{goal}\right|\right) + E\max_i \left|y_{n,i} - y_i^{start}\right|$ resulted in a large number of runs

ending up in local minima with the LPGA and poor quality results with the $\mu$GA. This situation was avoided by setting $\varphi$ to $= L\frac{\sqrt{\sum_i \left(y_i^{goal} - y_{n,i}\right)^2}}{\sqrt{\sum_i \left(y_i^{goal} - y_i^{start}\right)^2}}$. Interestingly, the opposite result was found in the case of shutdown. However, in the shutdown case, the $\mu$GA performed well with both functions of $\varphi$. In each operation mode, the TSGA presented no notorious difference in performance with either choice.

In both operation modes, it can be observed that the proposed $\mu$GA algorithm is capable of generating operating procedures that take less amount of time to reach the goal. This conclusion may not be apparent for the shutdown case in which the total operations time with $\mu$GA seems longer compared to that obtained with LPGA. The reason is that the proposed method takes an additional time to get closer to the goal, and reach a concentration of air of 0.9968 kg/kg that contrasts to the value of 0.9905 kg/kg obtained with LPGA. This was verified by examining the simulation results with $\mu$GA, indicating that the value of 0.9905 kg/kg (air) is reached at 1544 seconds, which shows that the proposed approach is faster.

The results also show that the proposed method presents better repeatability. This is particularly notorious in the shutdown case where three of the runs ended up trapped in local optima.

It can also be noted that the proposed method is more efficient in terms of computation time. The larger computation times for the shutdown are related to the fact that the length of the final solution (chromosome) of the shutdown case (67 genes) is much larger than the chromosome of the startup (19 genes). In average, each epoch took an average of 4.15 seconds for the startup and 13.73 seconds for the shutdown to complete.

## 5. Computational performance

To test the computational performance of the proposed method multiple runs were carried out for different problem sizes. Figure 9 shows the computational performance of the micro-GA algorithm for startup as a function of the problem size that results after increasing the number of possible valve positions from 2 to 9. This is also analogous to a fictitious increase of valves in the problem statement. The results show that the increase of CPU time is approximately linear, and all the problems can be solved in less than 2 minutes.

## 6. Conclusions

A method has been presented that uses a $\mu$GA algorithm to generate operating procedures for a mixing tank. The method is characterized by (1) the use of a good solution that is seeded into the initial population; (2) the use of chromosomes of variable length; and (3) the use of small population sizes. Numeric results show that the proposed method surpasses traditional genetic algorithms both in terms of quality of the solution and computational effort. In addition to the advantage in convergence speed, less memory is required to store the population.

One of the drawbacks of variable-length chromosomes is the possibility of bloat, that is the massive growth in the length of candidate solutions (Westerberg C. H., 2006). However, bloat does not occur in our case because the time-related term in the fitness function limits the chromosome's ability to grow.

As a note of caution, it is fair to mention that the proposed approach does not account for uncertainties associated with model parameters and constraints. There are uncertainties regarding the accuracy of the assumptions made about the flow rates, flammability constraints, and the valve operations.

Considering future research, one direction is the investigation of the parallelization of the inner loop of the $\mu$GA by considering the distribution of the elite solutions. It could also be useful to look at new algorithms with less parameters to be specified.

## Acknowledgements

This paper is a revised and extended version of a paper that was presented at the PSE 2012 conference. This work is dedicated to Prof. Yuji Naka on the occasion of his retirement as professor and head of the PSE Division at Tokyo Institute of Technology. The author is also indebted to the reviewers and editors of the Computers & Chemical Engineering journal for their time and effort, and for their comments that helped to improve the manuscript.

## References

Asprey, S. P., Batres, R., Fuchino, T., & Naka, Y. (1999). Optimal Simulation-based Operations Planning with Quantitative Safety Constraints. *Industrial and Engineering Chemistry Research, 36(6)*, pp. 2364-2374.

Barton, P. I., Banga, J. R., & Galán, S. (2000). Optimization of hybrid discrete/continuous dynamic systems. *Computers and Chemical Engineering, 24*, pp. 2171-2182.

Batres, R., Soutter, J., Asprey, S. P., & Chung, P. (2002). Operating procedure synthesis: science or art? *The Knolwedge Engineering Review, 17(3)*, pp. 261-294.

Benson, R., & Perkins, J. (1997). The future of process control - a UK perspective. *AiChE Symposium Series, Vol. 93, No. 316*, pp. 192-198.

Brie, A. H., & Morignot, P. (2005). Genetic planning using variable length chromosomes. *ICAPS 2005*, (pp. pp. 320-329).

Chapman, D. (1987). Planning for Conjunctive Goals. *Artificial Intelligence, 32(3)*, pp. 333-377.

Crawl, D. A. (2012). Minimize the Risks of Flammable Materials. *Chemical Engineering Progress, April*, pp. 28-33.

Davis, L. (1991). *Handbook of Genetic Algorithms.* United States: Van Nostrand Reinhold.

Flanagan, M. T. (2010). *Numerical Solution of Differential Equations*. Retrieved September 2012, from Michael Thomas Flanagan's Java Scientific and Numerical Library.: http://www.ee.ucl.ac.uk/~mflanaga/java/RungeKutta.html

Fusillo, R. H., & Powers, G. J. (1987). A Synthesis Method for Chemical Plant Operating Procedures. *Computers and Chemical Engineering, 11(4)*, pp. 369-382.

Galán, S., & Barton, P. I. (1997). Dynamic optimization formulations for operating procedure synthesis. *American Institute of Chemical Engineers Annual Meeting.* Los Angeles.

Glover, F. (1989). Tabu Search-Part I. *ORSA Journal on Computing, 1(3)*, pp. 190–206.

IChemE. (2006). *BP Process Safety Series: Safe Ups and Downs for Process Units (Sixth Edition).* IChemE.

ISA. (1995). *ANSI/ISA-S88.01 Batch Control Part 1:Models and Terminology.* USA: The International Society for Measurement and Control. .

Keedwell, E., & Khu, S.-T. (2005). A hybrid genetic algorithm for the design of water distribution networks. *Engineering Applications of Artificial Intelligence, 18*, pp. 461-472.

Krishnakumar, K. (1989). Microgenetic algorithms for stationary and nonstationary function optimization. *Proc. SPIE Vol. 1196, Intelligent Control and Adaptive Systems, Guillermo Rodriguez; Ed.*, pp. 289-296.

Kubota, N., Neya, H., & Taniguchi, K. (2002). Network and Evolutionary Programming for a Mobile Robot. *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02), CD-ROM*, pp. 119-123.

Kucharyson, R. (2006). Optimised procedural operations. *Petroleum Technology Quarterly, Q3*, pp. 93-101.

Lakshmanan, R., & Stephanopoulos, G. (1988). Synthesis of Operating Procedures for Complete Chemical Plants – II A Nonlinear Planning Methodology. *Computers and Chemical Engineering, 12(9/10)*, pp. 1003-1021.

Lee, J., Kim, S.-M., Park, H.-S., & Woo, B.-H. (2005). Optimum design of cold-formed steel channel beams using micro Genetic Algorithm. *Engineering Structures, 27*, pp. 17-24.

Matsumoto, M., & Nishimura, T. (1998). Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation 8*, pp. 3-30.

Rivas, R., Rudd, D. F., & Kelly, L. R. (1974). Computer-Aided Safety Interlock Systems. *AIChE Journal, 20(2)*, pp. 311-319.

Robbins, P. (1995). The use of a variable length chromosome for permutation manipulation in genetic algorithms. In *In D. Pearson,. N. Steele, and R. Albrecht, editors, Artificial Neural Nets and Genetic Algorithms* (pp. pp. 53–56). Springer-Verlag.

Rudnaya, S., Santosa, F., & Chiareli, A. O. (1998). Optimal design of a difractive optical element. Philadelphia.

Soutter, J. K., & Chung, P. W. (1995). Planning Architectures for Operating Procedure Synthesis. *Proceedings of the 8th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Gordon & Breach , Melbourne, Australia, 1995*, pp. 221-229.

Soutter, J., & Chung, P. W. (1996). Partial Order Planning with Goals of Prevention. *Proceedings of the 15th Workshop of the UK Planning & Scheduling Special Interest Group , 2, Liverpool John Moores University, School of Computing & Mathematical Sciences,*, pp. 300-311 .

Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V., & Ärzen, K. (1998). Automating operating procedure synthesis for batch processes: part I. Knowledge representation and planning framework. *Computers and Chemical Engineering, 22(11)*,

pp. 1673-1685.

Viswanathan, S., Johnsson, C., Srinivasan, R., Venkatasubramanian, V., & Ärzen, K. (1998). Automating operating procedure synthesis for batch processes: part II. Implementation and application. *Computers and Chemical Engineering, 22(11)*, pp. 1687-1698.

Weld, D. S. (1994). An Introduction to Least Commitment Planning. *AI Magazine, 15(4)*, pp 27-61.

Westerberg, C. H. (2006). *An Investigation into the use of Evolutionary Algorithms for Fully Automated Planning.* PhD Thesis, Artificial Intelligence Applications Institute, School of Informatics, University of Edinburgh.

Westerberg, C. H., & Levine, J. (2001). Optimising Plans using Genetic Programming. *Proceedings of the UK Workshop on Computational Intelligence (UKCI-01).* Edinburgh.

Yang, X., Li, K., & Xu, Q. (2011). Fine-tune ethylene unit startups: Advanced modeling methods provide useful information on operating main compressor system and feeds. *Hydrocarbon Processing, Volume 90, Issue 6*, pp. 73-78.

A**ppendix 1**. Micro genetic algorithms ($\mu$GAs)

Micro genetic algorithms ($\mu$GAs) are genetic algorithms characterized by small populations (typically no more than 10 individuals). As explained by Krishnakumar (1989), $\mu$GAs have been found to converge faster to the near-optimal solution. A $\mu$GA has an inner loop and outer loop. Within the inner loop, selection, and genetic operations are applied. In each iteration in the inner loop, the best-fit individual (elite solution) of each generation is kept in memory. Once convergence has been achieved, a new population is generated and the best-fit individual from the converged inner-loop is added to the population. In other words the best-fit individual is *seeded* or inserted into the initial population while the other individuals are randomly generated. In the $\mu$GA algorithm of Rudnaya *et al*. (1998), only selection and crossover are carried out at the inner loop while mutation is performed at the outer loop except on the best fit individual. A similar scheme is suggested by Lee *et al*. (2005). Due to the introduction of mutation operators that are specific for variable-length chromosomes, mutations in the proposed methodology are carried out in the inner loop and not in the outer loop.

A**ppendix 2**. Parameter values used in the numerical experiments.

| Parameters | Startup | Shutdown |
|---|---|---|
| Initial chromosomes size | 25-30 | 45-55 |
| Crossover probability | 0.8 | 0.8 |
| Elitist crossover probability | 0.7 | 0.5 |
| Crossover and mutate probability | 0.5 | 0.5 |
| Mutation probability | 0.0008 | 0.08 |
| Shrink mutation probabiliy | 0.25 | 0.4 |
| Grow mutation probabiliy | 0.08 | 0.08 |
| Swap mutation probability | 0.008 | 0.008 |
| Parameter change mutation probability | 0.1 | 0.1 |
| $w_{\max}$ | 0.01 | 0.0001 |
| $w_2$ | $1 - w_{\max}$ | $1 - w_{\max}$ |
| $m$ | 0.0 | 0.01 |
| $a$ | 50.0 | 50.0 |
| $b$ | 50.0 | 50.0 |
| $D$ | 0 | 2.0 |
| $E$ | 0 | 18.0 |
| $F$ | $1 \times 10^9$ | $1 \times 10^9$ |
| $L$ | 15 | 0.0 |
| $A$ | 2.0 | 2.0 |
| $B$ | 18.0 | 18.0 |
| $N$ | 40 | 40 |

Table 1. Average and best values for startup. Values corresponding to the best run are shown in parenthesis.

Table 2. Average and best values for shutdown. Values corresponding to the best run are shown in parenthesis.

Figure 1. The mixing vessel.

Figure 2. The proposed methodology.

Figure 3. Example of an operations chromosome.

Figure 4. Comparison of the startup operation trajectories between the proposed method and LPGA.

Figure 5. Comparison of the valve profiles for the startup case between the proposed method and LPGA.

Figure 6. Comparison of the shutdown operation trajectories between the proposed method and LPGA.

Figure 7. Comparison of the valve profiles for the shutdown case between the proposed method and LPGA.

Figure 8. Convergence of LPGA and the $\mu$ GA

Figure 9. CPU time versus problem size.

**Table 1**

Table 1. Average and best values for startup. Values corresponding to the best run are shown in parenthesis.

| | | Traditional GA | Seeded GA | $\mu$GA |
|---|---|---|---|---|
| Computation time [s] | | 609.3321 (439.282) | 113.9683 (168.5310) | 80.5708 (76.1150) |
| Total operations time [s] | | 523.0909 (483) | 488.6 (459) | 481.2 (450) |
| Final State | Steam [kg/kg] | 0.1012 (0.0999) | 0.1007 (0.1011) | 0.1017 (0.1005) |
| | Propylene [kg/kg] | 0.1498 (0.1508) | 0.1499 (0.1494) | 0.1505 (0.1505) |
| | Air [kg/kg] | 0.7490 (0.7493) | 0.7493 (0.7494) | 0.7478 (0.7490) |
| Mean squared concentration error | | 0.0021 (0.0011) | (0.0025) (0.0014) | 0.0038 (0.0012) |
| Fitness | | 188.1390 (204.8079) | 199.8628 (214.7448) | 202.7944 (219.4857) |

**Table 2**

Table 2. Average and best values for shutdown. Values corresponding to the best run are shown in parenthesis.

| | | Traditional GA | Seeded GA | $\mu$GA |
|---|---|---|---|---|
| Computation time [s] | | 1260.7870 (1241.3440) | 1813.1100 (1821.9110) | 277.4407 (276.366) |
| Total operations time [s] | | 1274.1 (1548.0) | 1885.5 (1938.0) | 1920.0 (1920.0) |
| Final State | Steam [kg/kg] | 0.1034 (0.008687) | 0.0038 (0.0032) | 0.0030 (0.0030) |
| | Propylene [kg/kg] | 0.01025 (0.0007670) | 0.0003010 (0.0002464) | 0.000221 (0.000221) |
| | Air [kg/kg] | 0.8863 (0.9905) | 0.9958 (0.9965) | 0.9968 (0.9968) |
| Mean squared concentration error | | 0.15410 (0.01286) | 0.005637 (0.0047) | 0.0044 (0.0044) |
| Fitness | | 634.2825 (721.3214) | 726.4969 (727.1368) | 711.3784 (711.3784) |

**Figure 1**

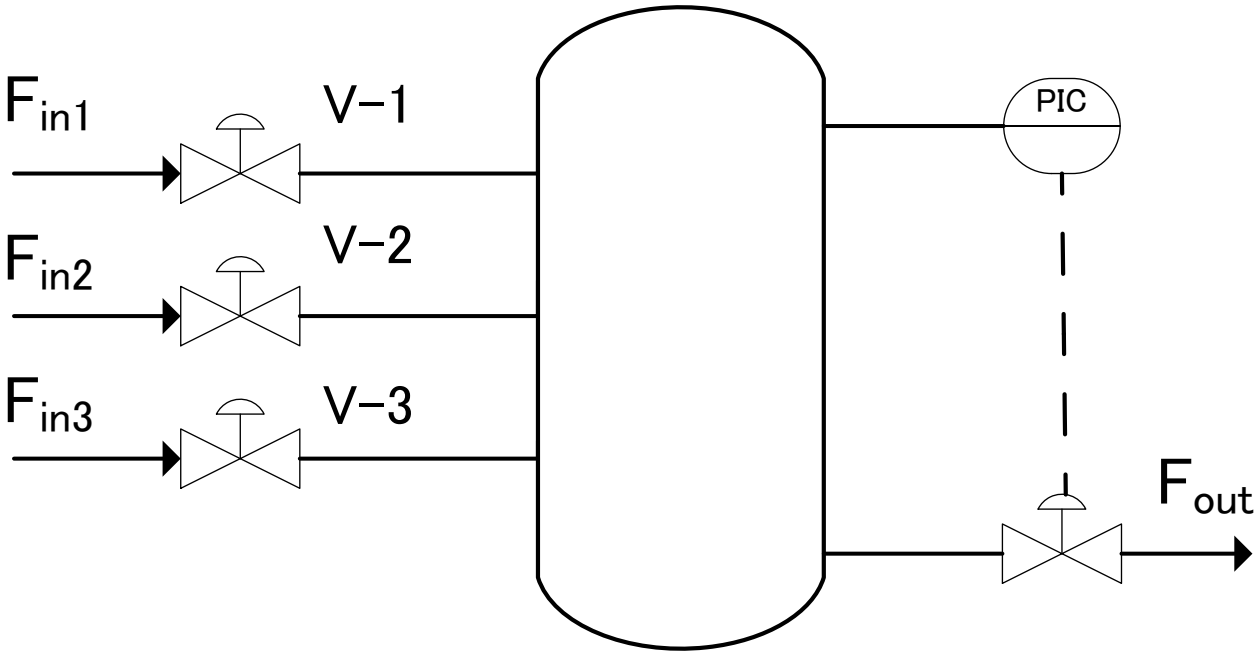


**Figure 2**

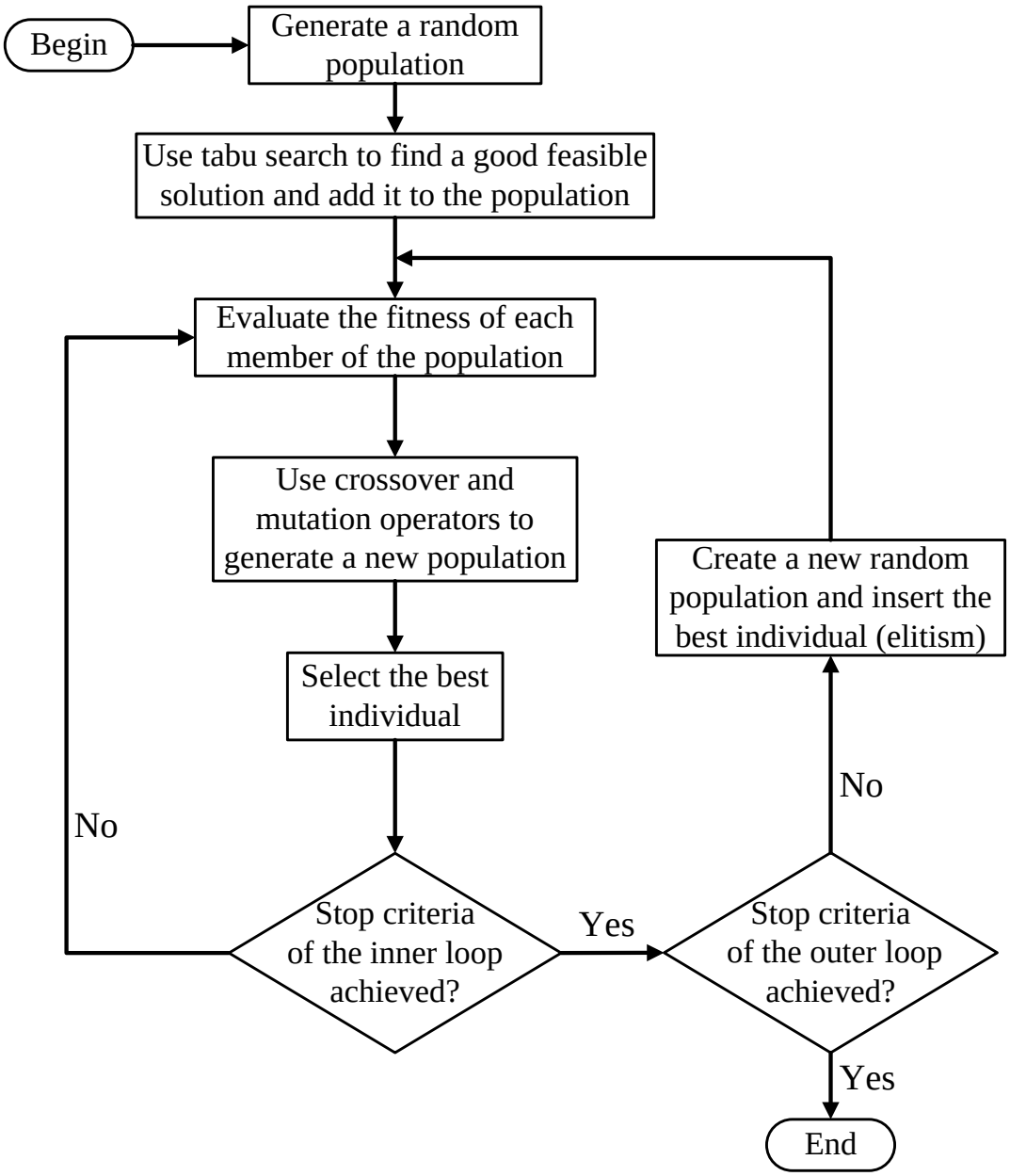

Begin
Generate a random population
Use tabu search to find a good feasible solution and add it to the population
Evaluate the fitness of each member of the population
Use crossover and mutation operators to generate a new population
Select the best individual
Stop criteria of the inner loop achieved?
No
Yes
Stop criteria of the outer loop achieved?
No
Create a new random population and insert the best individual (elitism)
Yes
End

**Figure 3**

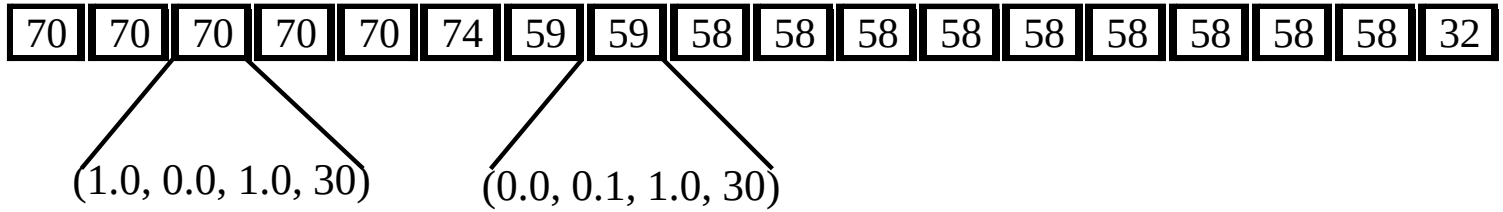

70 70 70 70 70 74 59 59 58 58 58 58 58 58 58 58 58 32
(1.0, 0.0, 1.0, 30)
(0.0, 0.1, 1.0, 30)

**Figure 4**

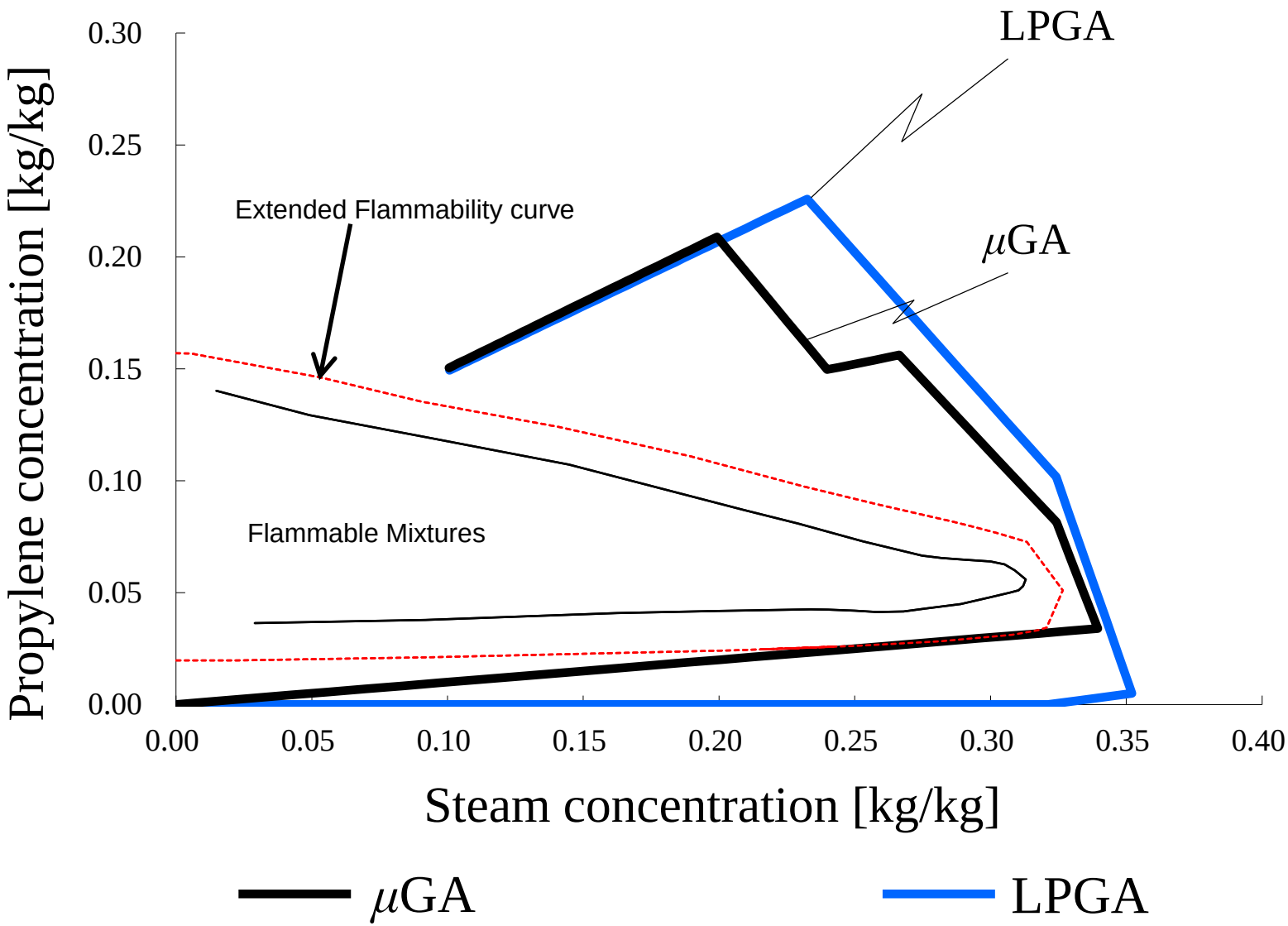

LPGA
μGA
Extended Flammability curve
Flammable Mixtures
Propylene concentration [kg/kg]
Steam concentration [kg/kg]
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
μGA
LPGA

**Figure 5**

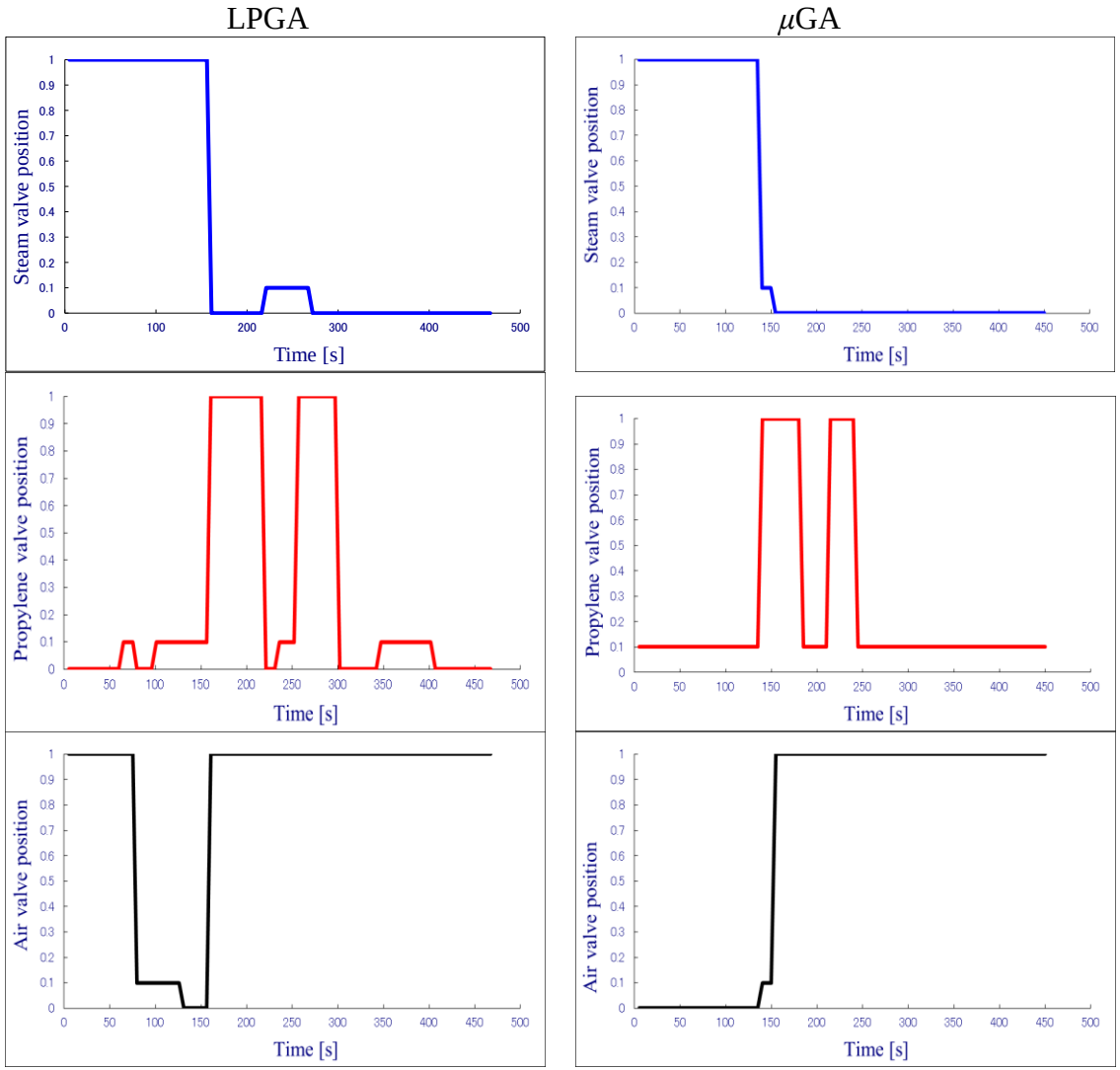

LPGA
μGA
Steam valve position
Propylene valve position
Air valve position
Time [s]

**Figure 6**

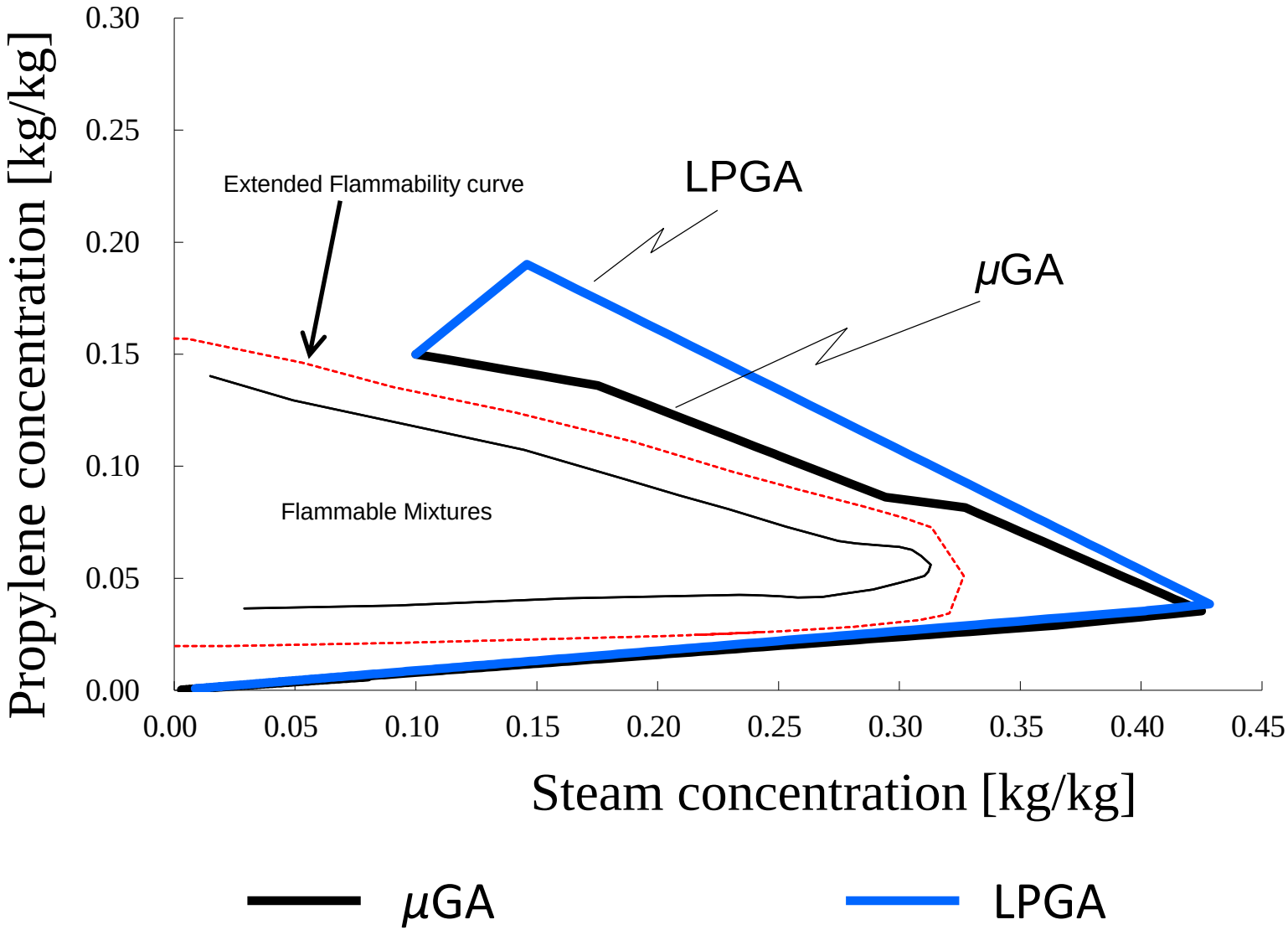

Propylene concentration [kg/kg]
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Extended Flammability curve
LPGA
μGA
Flammable Mixtures
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
0.45
Steam concentration [kg/kg]
μGA
LPGA

**Figure 7**

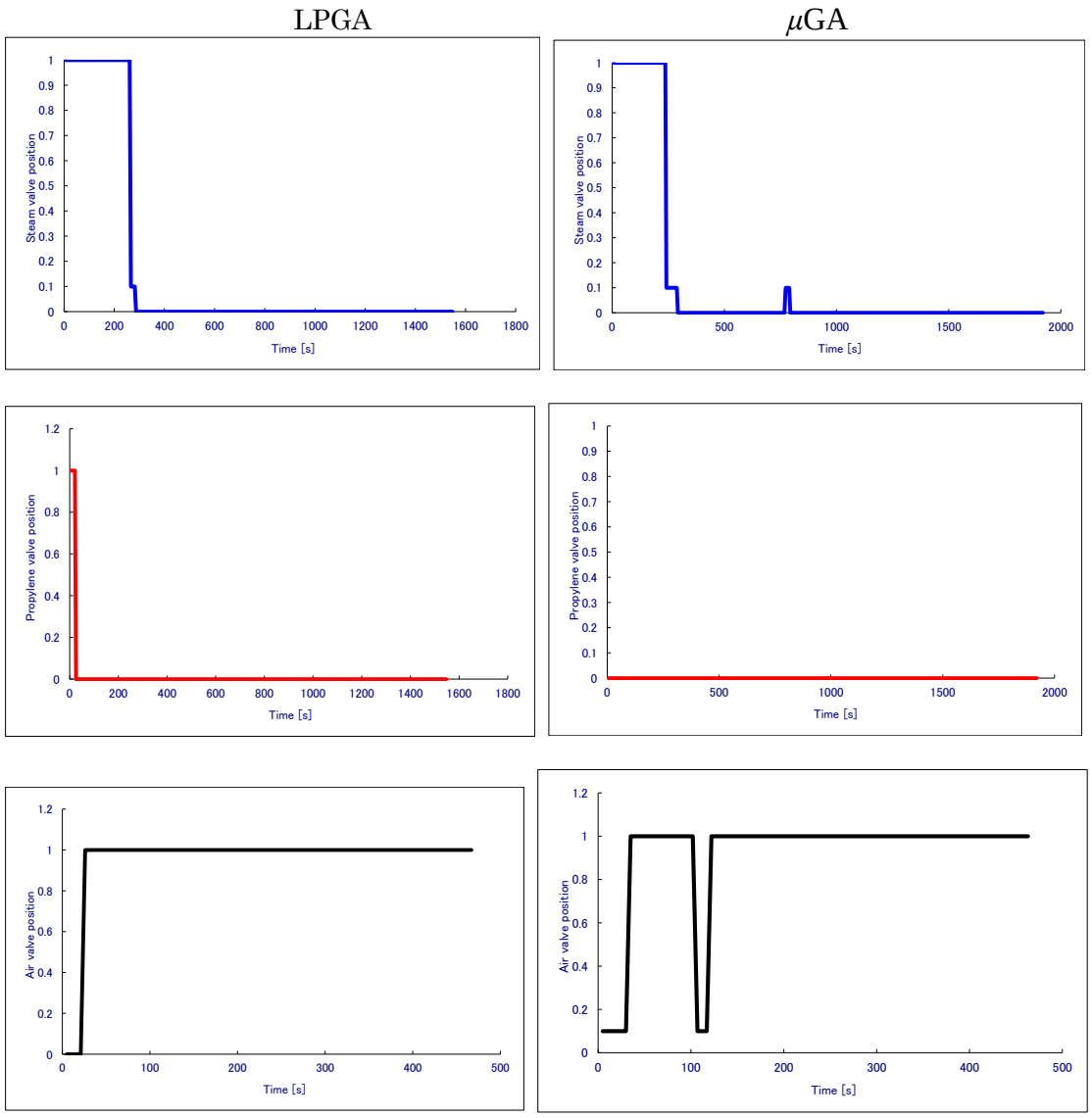
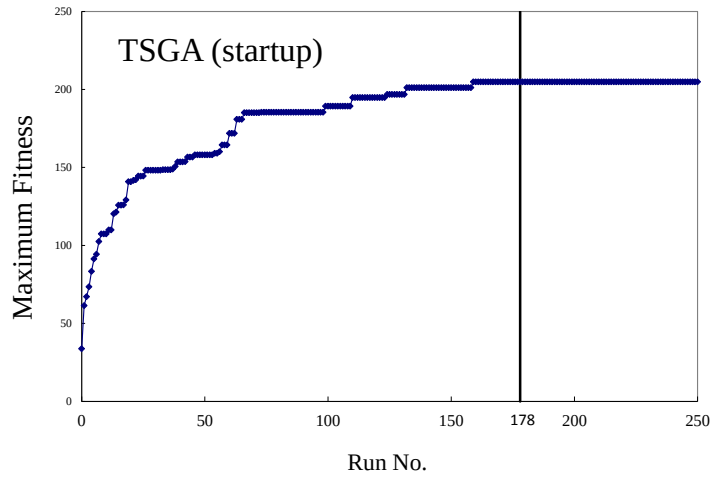
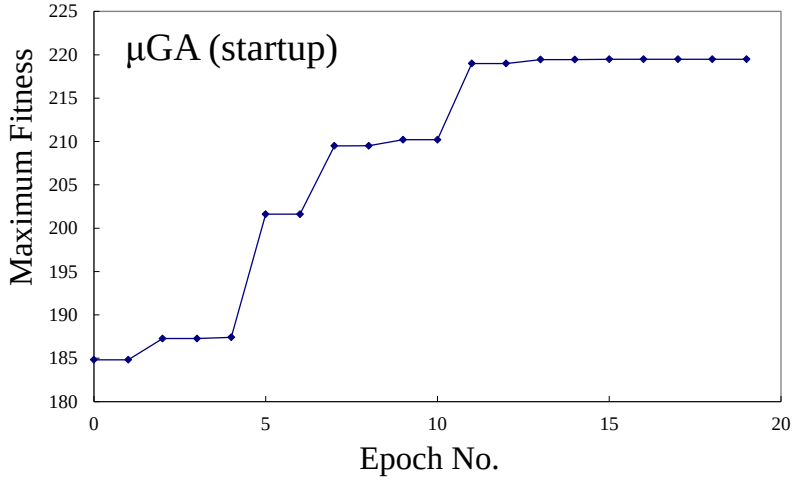
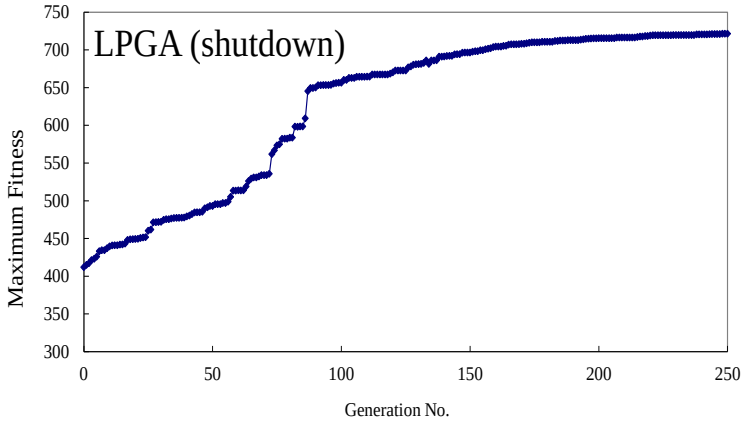

LPGA
μGA
Steam valve position
Time [s]
Propylene valve position
Time [s]
Air valve position
Time [s]

**Figure 8**


TSGA (startup)
Maximum Fitness
Run No.
0
50
100
150
178
200
250


μGA (startup)
Maximum Fitness
Epoch No.
180
185
190
195
200
205
210
215
220
225
0
5
10
15
20


LPGA (shutdown)
Maximum Fitness
Generation No.
300
350
400
450
500
550
600
650
700
750
0
50
100
150
200
250

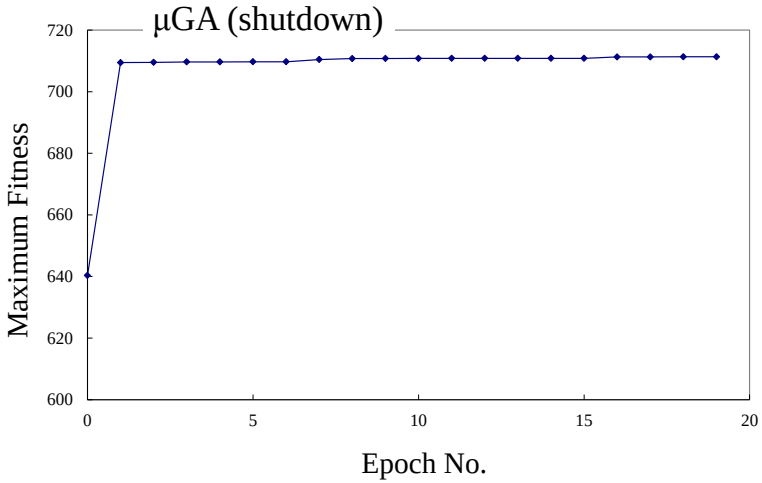
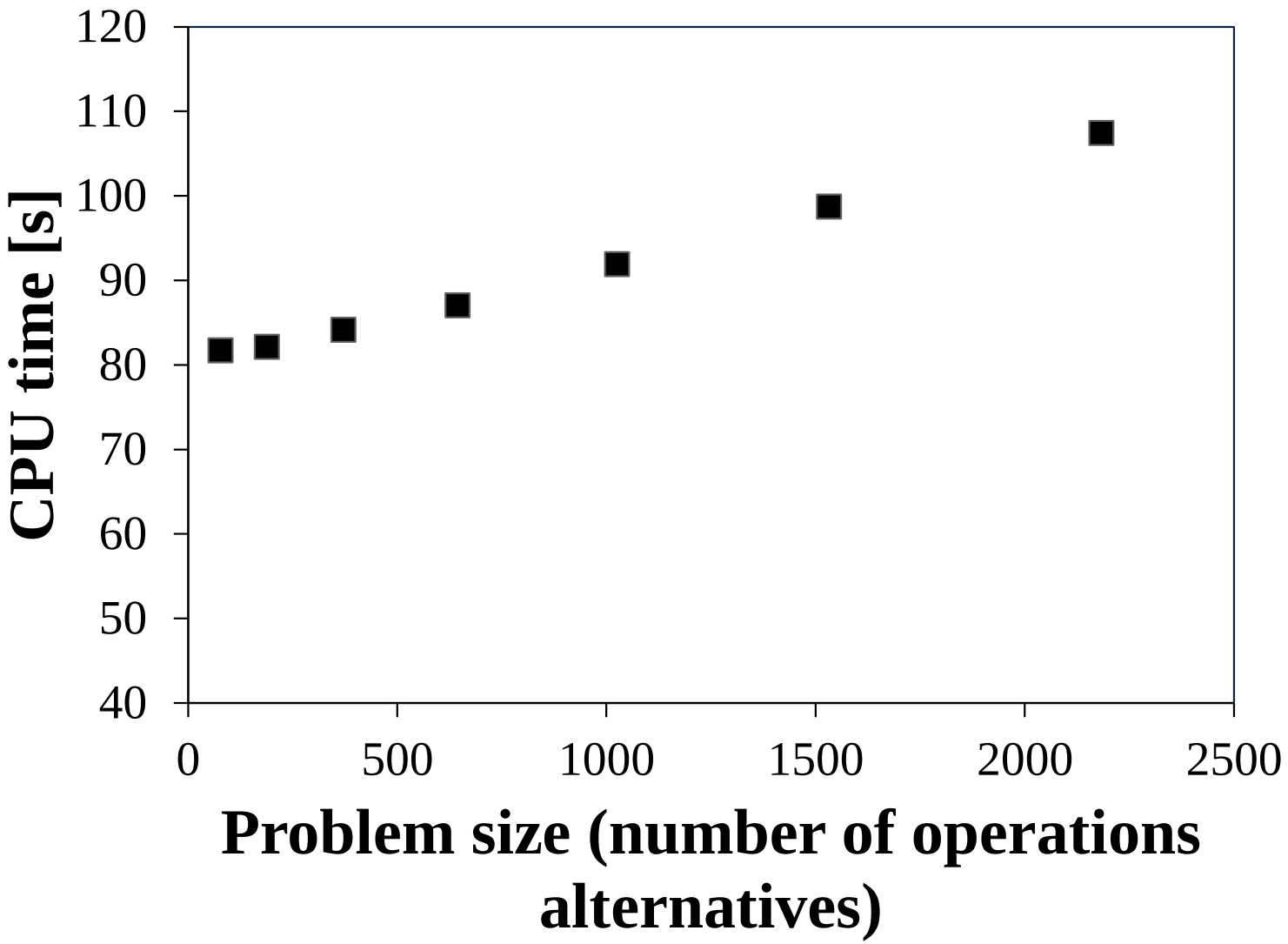

μGA (shutdown)
Maximum Fitness
Epoch No.
600
620
640
660
680
700
720
0
5
10
15
20

Figure 9


120
110
100
90
80
70
60
50
40
CPU time [s]
0
500
1000
1500
2000
2500
Problem size (number of operations alternatives)