Korean J. of Chem. Eng., **14**(4), 225-232 (1997)

A HYBRID SYSTEM OF HEURISTICS AND SIMULATED ANNEALING FOR LARGE SIZE SCHEDULING PROBLEMS OF BATCH PROCESSES WITH PIPELESS INTERMEDIATE STORAGE

Hyung Joon Kim, Jae Hak Jung†, Minseok Kim* and In-Beum Lee*

School of Chemical Engineering & Technology, Yeungnam University, 214-1, Dae-dong, Kyongsan 712-749, Korea
*Automation Research Center, Pohang University of Science and Technology, San 31, Hyoja-dong, Pohang 790-784, Korea
(Received 7 September 1996 • accepted 23 July 1997)

Abstract – This paper describes a new scheduling solution for large number multi-product batch processes with complex intermediate storage system. Recently many batch chemical industries have turned their attention to a more efficient system known as a pipeless batch system. But existing plants need to change their systems to pipeless systems, piece by piece. In this case, current systems are changed to pipeless systems by way of non critical process operations such as through the use of intermediate storage. We have taken the conventional batch plant with a pipeless storage system as an objective process. Although the operation of a pipeless storage system becomes more complex, its efficiency is very high. With this system, all of the storage should be commonly used by any batch unit. For this reason, solving the optimal scheduling problem of this system with a mathematical method is very difficult. Despite the attempts of many previous researches, there has been no contribution which solves the scheduling of intermediate storage for complex batch processes. In this paper, we have developed a hybrid system of heuristics and Simulated Annealing (SA) for large multi-product processes using a pipeless storage system. The results of this study show that the performance and computational time of this method are superior to that of SA and Rapid Access Extensive Search (RAES) methods.

Key words: *A Hybrid System of Heuristics and Simulated Annealing, Pipeless Intermediate Storage, Metropolis Algorithm, Probability of Up-Hill Movement*

INTRODUCTION

As the trend toward multi-product batch processes has increased with the Chemical Process Industry (CPI), multi-product batch scheduling has been actively studied. In batch process operations, there are various methods of operation for increasing the productivity. One method is adjusting the intermediate storage policies. The different types of intermediate storage policies which have been frequently studied are unlimited intermediate storage (UIS), finite intermediate storage (FIS), no intermediate storage (NIS), zero wait (ZW) and mixed intermediate storage (MIS) policies.

Recently, Ku and Karimi [1990] proposed a new method of using intermediate storage tanks, called a shared storage system, as a block of the MIS policy. Jung et al. [1996] have proposed the common intermediate storage (CIS) policy, in which some storage is shared throughout the whole system to accomplish a more complete flexibility. It has been accepted as a highly efficient intermediate storage policy.

There are also many contributions for the scheduling of multi-product batch processes with several intermediate storage policies. A major difficulty is that optimal scheduling problems for multi-product batch processes are NP-complete. Over the past decade, two kinds of methodologies for solving these kinds of scheduling problems have been developed. One method is based on well known mathematical optimization techniques, such as Mixed Integer Linear Programming (MILP) and Mixed Integer Non-Linear Programming (MINLP). The other method is heuristic or rule based, like branch and bound (BAB) or some application of artificial intelligence.

Recently, Ku and Karimi [1991] have studied SA as a new method which is a type of random search using a monte-carlotic algorithm. This method has improved the results of several types of scheduling problems.

Jung et al. [1994] also have developed a Modified Simulated Annealing Approach (MSA) which is composed of a two stage search algorithm. It has provided excellent results for scheduling problems with UIS and NIS policies, it has not been applied to the batch scheduling of the complicated intermediate storage policy, for example, CIS.

Since the pipeless system for the batch chemical plant was introduced, research and application of pipeless systems in industry have increased. Pipeless systems are being considered, not only for grassroots plants but also for the redesign of existing plants. Usually, the objective of the latter case is to achieve a partial improvement of efficiency before constructing whole pipeless batch plant. In these cases, designers would like to apply pipeless systems to non critical parts of the existing processes. One of the process which is non critical for operation is the intermediate storage system. Moreover, the efficiency of the shared pipeless storage system is very high and it achieves a great improvement in produc-

†Author to whom all correspondence should be addressed.

tivity [Jung et al., 1996].

The scheduling of batch chemical processes under pipeless intermediate systems with MILP or MINLP is difficult to solve because of its complicated operation. Most industries have more than 20 products. As the problem size increases the difficulty in solving the problem increases exponentially.

In this paper we have developed a hybrid system using heuristics and SA for large multi-product batch chemical processes with a pipeless intermediate storage system. For pipeless systems, the transfer time of materials is much shorter than conventional pipe-valve systems. We can define the transfer time of materials as that of vessel time from dispatching to charging. The transfer time of a pipeless vessel varies with the length of vessel moving, so we can fix the transfer time. The price of the vessel is relatively low so it can be considered an UIS type. But for considering more actual system, we are taking into account that the number of vessels is a given finite number. After setting all the assumptions of the pipeless intermediate storage, we solved the problems using one or two moving vessels for intermediates. We also solved the same problems using RAES and SA of Ku and Karimi [1991] with the same parameters. Four hundred randomly generated problems were solved and the results of this study outperformed RAES and SA.

CONVENTIONAL CHEMICAL BATCH PLANT WITH PIPELESS MOVABLE STORAGE

Even in a flowshop batch plant the ordered product items can be changed. In case of changing the ordered product items, the operations which require specific intermediate storage policies, i.e., NIS, FIS and ZW modes, can be changed in the same multi-batch system. In order to meet this requirement of flexible operation, Jung et al. [1996] suggest the CIS intermediate storage system. An example of the system diagram of CIS with conventional pipe-valve batch processes of fixed storage is shown in Fig. 1.

The conventional CIS system has many pipelines for storage and may cause an overcharge capital costs as well as have a high probability of creating faults. The system suggested in this paper is simpler, with pipeless storage vessels which move between storage, set-up site and units to transfer the materials. It is not a type of full scale pipeless plant but a partial use of the pipeless concept, i.e., only storage facilities are operated as pipeless moving vessels. Materials

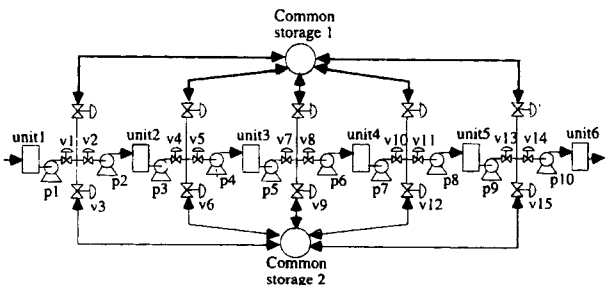

Fig. 1. Conventional pipe-valve batch process with fixed storages.

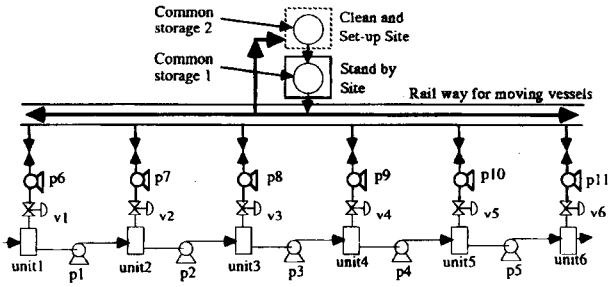

Fig. 2. Conventional batch process with pipeless movable storage system.

are still transferred by pipes and pumps from unit to unit. The schematic diagram of the pipeless type of intermediate storage for a batch plant is shown in Fig. 2.

For this system, a new transfer time between storage and the unit should be defined. Since the transfer time between storage and unit of pipeless intermediate storage system is the sum of times of moving storage, the total transfer time may become shorter than that of conventional pipe-valve and pump storage system. This is due to the net material transfer time being overwhelmingly reduced. Moreover, the pipeless system can greatly reduce the waste material generated by the cleaning of pipelines and valve-pumps. This system also completes the highly flexible operation of intermediate storage as the CIS system.

We can guess two possible types of pipeless intermediate storage systems. One type uses unlimited number of moving vessels and the other uses limited number of moving vessels. We have focused on the latter case. The completion time of the pipeless storage system has also been proven with same procedures of the CIS system of Jung et al. [1996].

Similarly to the CIS system, the completion time of the pipeless movable storage system is not longer than that of the NIS system [Eq. (2)] and not shorter than that of the UIS system [Eq. (1)].

$$C_{ij}\ (\text{UIS}) = \max\left[C_{(i-1)j},\ C_{i(j-1)}\right] + t_{ij} \tag{1}$$

$$C_{ij}\ (\text{NIS}) = \max\left[C_{(i-1)j}, C_{i(j-1)},\ C_{(i-1)(j+1)} - t_{ij}\right] + t_{ij} \tag{2}$$

When one of the common storage vessels is vacant, the completion time is given by Eq. (1). If no storage vessel is available, the completion time is given by Eq. (2). If one of the storage vessels is partially available, the completion time varies from Eq. (1) to the value of $C_{(i-1)(j+1)}$. So storage starting time $SS(l,k)$ and storage ending time $SE(l,k)$ for product l after unit k should be known. For product l after unit k, the time when one of the storage vessels is about to reach a usable state (i.e., beginning of the storage ending time) is earlier than the completionime of the NIS policy for i product at j unit, then the completion time for pipeless movable storage system ($C_{ij}(pl)$) becomes ($SE(l,k)$), otherwise $C_{ij}(pl)$ is given by Eq. (2).

Assuming that transfer and setup times are zero to make the problem simpler, we can describe the system as follows.

If $C_{ij}(\text{UIS}) > C_{(i-1)(j+1)}$ or at least one vessel is usable then $C_{ij}(pl)$=Eq. (1).

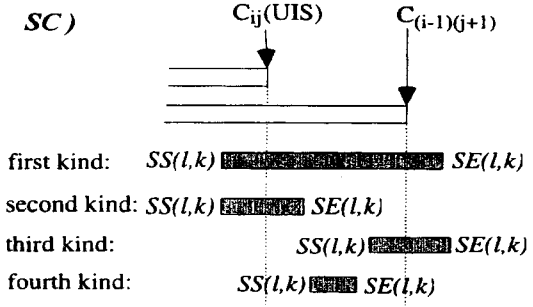

Fig. 3. Four kinds of storage using pattern of one common storage without transfer and set up times.

Else, if no vessel is usable then $C_{ij}(pl)$=Eq. (2).

Else, if some vessel is partially usable then we need storage using the time for minimum completion as the time of the process and we define this situation as the storage checking state (sc).

We have suggested a simpler method than the existing one to check the availability of storage vessels. For an estimation of the above mentioned sc state, we used a one step comparison of the value of $SS(l,k)$, Eq. (1) and $C_{(i-1)(j+1)}$.

For the sc state, four kinds of storage using patterns are possible in the storage vessel as shown in Fig. 3.

The storage using times that are concerned with the interval from Eq. (1) to Eq. (2) can be selected by the following conditions:

$SS(l,k) < C_{(i-1)(j+1)}$ (condition 1)

$SE(l,k) >$ Eq. (1) (condition 2)

If both condition 1 and condition 2 are satisfied, the situation of storage should be considered as the sc state. We can compress the above four type into one equation for the sc state.

$$C_{ij}(pl) = \min [C_{ij}(\mathrm{NIS}), SE(l,k)] \quad (3)$$

Rarely do the second, third, and fourth kind of storage using patterns arise in one storage vessel sequentially. Thus, Eq. (3) should be modified as follows:

$$C_{ij}(pl) = \min [C_{ij}(\mathrm{NIS}), \max\{SE^*(l,k)\}] \quad (4)$$

definition 1) $SE^*(l,k)$ is a set of storage ending (emptying) time ($SE^*(l,k)$s) which satisfies both condition 1 and condition 2 for product i after unit j.

If the number (P) of storage vessels is a multiple (P>1), then Eq. (4) should be modified again as follows:

$$C_{ij}(pl) = \min[C_{ij}(\mathrm{NIS}), \max\{SE_1^*(l,k)\}, \max\{SE_2^*(l,k)\}, \cdots, \max\{SE_p^*(l,k)\}] \quad (5)$$

definition 2) $SE_p^*(l,k)$ is a set of storage ending (emptying) time ($SE^*(l,k)$s) of the pth (p=1,2,···,P) storage vessel which satisfies both condition 1 and condition 2 for product i after unit j.

We additionally suggested the algorithm to define $SS_p(i,j)$ and $SE_p(i,j)$ (i=1, ···, N, j=1, ···, M) as a merging form in the recursive completion time determination algorithm.

For given i and j:
  If (C_{ij}(UIS) < $C_{(i-1)(j+1)}$) then
    Do (for every introduced intermediate storage p (p=1,2, ···,P))
      If (there is no [$SS_p(l,k)$, $SE_p(l,k)$] to satisfy the condition
      $SS_p(l,k) < C_{(i-1)(j+1)}$ and $SE_p(l,k) > C_{ij}$(UIS)) then
      $SS_p(i,j) = C_{ij}$(UIS)
      $SE_p(i,j) = C_{(i-1)(j+1)}$
      Else
        If (there are one or more $SE_p(l,k)$s to satisfy $SE_p(l,k) < C_{(i-1)(j+1)}$) then
        $SS_p(i,j) = \max(SE_p(l,k))$ (l=1, ···, N, k=1, ···, M)
        $= \max(SE_p^*(l,k))$
        $SE_p(i,j) = C_{(i-1)(j+1)}$
        Else
        $SS_p(i,j) = 0$
        $SE_p(i,j) = 0$ and unit j should be holding the product i until the next unit j+1 becomes ready to process the product i
      End if
  End if
Continue
  Else
    for every introduced intermediate storage p(p=1,···,P)
    $SS_p(i,j) = 0$
    $SE_p(i,j) = 0$ this block can be eliminated by the initial definition that every $SS_p(i,j)$ and $SE_p(i,j)$ is zero
  End if

From all the above explanations, we can compose a completion time algorithm with zero transfer and setup times of a pipeless storage system from the algorithm which defines $SS_p(i,j)$ and $SE_p(i,j)$.

$$C_{ij}(pl) = \max[C_{(i-1)j}, C_{i(j-1)}, \min\{\max(SE_1^*(l,k)), \max(SE_2^*(l,k)), \cdots, \max(SE_p^*(l,k)), C_{(i-1)(j+1)}\} - t_{ij}] + t_{ij} \quad (6)$$

$SE_p^*(l,k)$: every $SE_p(l,k)$ (l=1, ···, N, k=1, ···, M) for storage p which satisfies both condition 1 and condition 2.

We defined a_{ij} as the transfer time which is required for transferring product i out of unit j to (j+1), and $S_{i(i+1)j}$ as the set-up time which is required for preparing the (i+1)th prod-

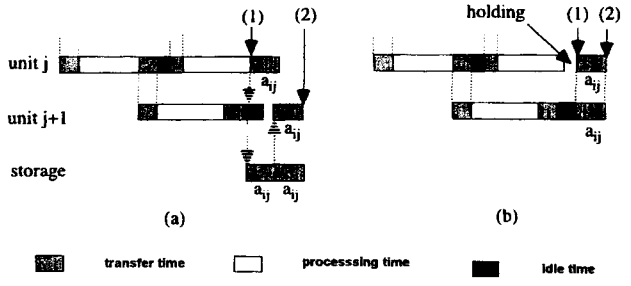

Fig. 4. Comparison of holding time in unit and used storage for some special cases.

uct after the i th product with a constant value. Additionally, as(l,k) should be defined as the storage clean-up and preparing time after using of l th product after k unit. We defined C_{ij} as the completion time of processing the i th product at the j th unit. In actual operation, the completion time must include the processing time and the time of completely transferring out the i th product from j th unit. So we defined $C^*_{ij}=C_{ij}+a_{ij}$ for the true sense of the completion time of i th product at j unit.

When zero transfer and set-up times are assumed and one of the storage vessels is available, the materials within the units can always use the storage and $C_{ij}(pl)$ is equal to Eq. (1). For the case of considering non-zero transfer and set-up times as shown in Fig. 4, although one of the storages is available, the situation can occur that holding the product in the unit is better than using the available storage since this doubles the transfer time when using storage.

For unit j in Fig. 4, using storage operation (a) is better than product holding in unit (b), but for unit (j+1), operation (b) is better than operation (a). Consequently we did not know which alternative was the better choice. Operation and calculation of the completion time of operation (a) is slightly simpler than operation (b) because the storage is always used if one of the storage vessels is available when $C_{ij}(\text{UIS})<C_{(i-1)(j+1)}$. We named them 'operation (a)' and 'operation (b)' as mentioned above.

To consider non-zero transfer and set-up times, C_{ij}(UIS) and C_{ij}(NIS) are redefined as Eq. (7) and Eq. (8). We have introduced a new variable E_{ij}, that is $C_{ij}+a_{ij}+S_{i(i+1)j}$.

$$C_{ij}(\text{UIS})=\max[C_{(i-1)j}+a_{(i-1)j}+S_{(i-1)ij},\ C_{i(j-1)}]+t_{ij}+a_{i(j-1)} \tag{7}$$

$$C_{ij}(\text{NIS})=\max[C_{(i-1)j}+a_{(i-1)j}+S_{(i-1)ij},\ C_{i(j-1)},\ C_{(i-1)(j+1)}+a_{(i-1)(j+1)} +S_{(i-1)i(j+1)}-t_{ij}-a_{i(j-1)}]+t_{ij}+a_{i(j-1)} \tag{8}$$

The determination of completion time is similar with the case of zero transfer and set-up times except for the sc state procedure. Considering four kinds of partial use of storage vessels is slightly complicated. The storage use times of the products that are concerned with the interval from Eq. (7) to $E_{(i-1)(j+1)}$ (i.e., ith product needs storage) can be selected using the following conditions:

$$SS(l,k) < E_{(i-1)(j+1)}+a_{ij}+as(i,j) \qquad \text{(condition 3)}$$

$$SE(l,k)+as(l,k) > \text{Eq. (7)} \qquad \text{(condition 4)}$$

Using a similar reason as for the cases of zero transfer and set-up times, Eq. (9) with definition 2 can be composed for sc state.

$$C_{ij}(pl)=\min[E_{(i-1)(j+1)},\ \max\{SE^*_1(l,k)+as(l,k)\},\ \max\{SE^*_2(l,k)+as(l,k)\},\ \cdots,\ \max\{SE^*_p(l,k)+as(l,k)\}] \tag{9}$$

It is assumed that the type of every storage vessel (p=1, $\cdots$,P) introduced is the same, so as(i,j) for each storage is equal.

At this point, the storage starting and ending times are required. They are determined in a similar way as in the algorithm mentioned above.

Given i and j:

If (C_{ij}(UIS) < $E_{(i-1)(j+1)}$ (for operation(a)), C_{ij}(UIS)+a_{ij} < $E_{(i-1)(j+1)}$ (for operation (b)) then
Do (for every introduced intermediate storage p (p=1,2,$\cdots$P))
If (there is no [$SS_p(l,k)$, $SE_p(l,k)$] to satisfy both condition 1 and condition 2) then
  $SS_p(i,j)=C_{ij}$(UIS)
  $SE_p(i,j)=E_{(i-1)(j+1)}+a_{ij}$
Else
  If (there are one or more ($SE_p(l,k)$+as(l,k))s to satisfy $SE_p(l,k)+as(l,k) < E_{(i-1)(j+1)}-a_{ij}$ then
  $SS_p(i,j)=\max(SE_p(l,k)+as(l,k))$ (l=1, $\cdots$, N, k=1, $\cdots$, M)
  $=\max(SE^*_p(l,k)+as(l,k)$
Else
  $SS_p(i,j)=0$
  $SE_p(i,j)=0$ and unit j should be holding product i until the next unit (j+1) becomes ready to process the product i
  End if
End if
Continue
Else
  for every intermediate storage p(p=1,$\cdots$,P) introduced
  $SS_p(i,j)=0$
  $SE_p(i,j)=0$ this block can be eliminated by the initial definition stating that every $SS_p(i,j)$ and $SE_p(i,j)$ is zero
End if

For operation (a):

$$C_{ij}(pl)=\max[E_{(i-1)j},\ C_{i(j-1)},\ \min\{\max(SE^*_1(l,k)+as(l,k)),\ \max(SE^*_2(l,k)+as(l,k)),\ \cdots,\ \max(SE^*_p(l,k)+as(l,k)),\ E_{(i-1)(j+1)}\}-t_{ij}-a_{i(j-1)}]+t_{ij}+a_{i(j-1)} \tag{10}$$

For the case of operation (b):

$$C_{ij}(pl)=\max[\max\{E_{(i-1)j},\ C_{i(j-1)},\ E_{(i-1)(j+1)}-t_{ij}-a_{i(j-1)} (\max[E_{(i-1)j},\ C_{i(j-1)}]+a_{i(j-1)} > E_{(i-1)(j+1)}-t_{ij}-a_{i(j-1)}),\ \min\{\max(SE^*_1(l,k)+as(l,k)),\ \max(SE^*_2(l,k)+as(l,k)),\ \cdots,\ \max(SE^*_p(l,k)+as(l,k)),\ E_{(i-1)(j+1)}\}-t_{ij}-a_{i(j-1)}]+t_{ij}+a_{i(j-1)} \tag{11}$$

$SE^*_p(l,k)$: every $SE_p(l,k)$ (l=1, $\cdots$, N, k=1, $\cdots$, M) for storage p which satisfies both condition 3 and condition 4.

At the final stage, we can combine the completion time algorithm for the ZW block by Jung et al. [1994] and the earlier mentioned completion time algorithm of Eq. (10) and (11). Finally, we determine the completion times of the products in a pipeless storage system with a ZW block extending from unit Q to R without transfer and set-up times. For products i=1,2,$\cdots$,N and operation (a) :

$$C_{ij}(pl)=\max[E_{(i-1)j},\ C_{i(j-1)},\ \min\{\max(SE^*_1(l,k)+as(l,k)),\ \max(SE^*_2(l,k)+as(l,k)),\ \cdots,\ \max(SE^*_p(l,k)+as(l,k)),\ E_{(i-1)(j+1)}\}-t_{ij}-a_{i(j-1)}]+t_{ij}+a_{i(j-1)} \quad (\text{for } j=1,\ 2,\ \cdots,\ Q-1 \text{ and } R+1,\ R+2,\ \cdots M) \tag{12}$$

$$C_{iQ}(pl)=\max\left[C'_{iR}-\sum_{j=Q}^{R}t_{ij}-\sum_{j=Q}^{R}a_{ij},\ C_{i(Q-1)}\right]+t_{iQ}+a_{i(Q-1)} \quad (\text{for } j=Q) \tag{13}$$

$$C_{ij}(pl)=C_{i(j-1)}+t_{ij}+a_{i(j-1)} \qquad (\text{for } j=Q+1,\cdots,R-1) \tag{14}$$

$$C_{iR}(pl)=\max[C_{i(R-1)},\ \min\{\max(SE^*_1(l,k)+as(l,k)),$$

$$\max(SE_2^*(l,k)+as(l,k)), \cdots,$$
$$\max(SE_p^*(l,k)+as(l,k)), E_{(i-1)(j+1)}-t_{ij}-a_{i(j-1)}]+t_{ij}+a_{i(j-1)}$$
(for $j=R$) (15)

$$C_{ij}^*(pl)=C_{ij}(pl)+a_{ij} \quad (16)$$

where, $C'_{iR}=\max\left[C_{(i-1)Q}+S_{(i-1)iQ}+\sum_{j=Q}^{R}t_{ij}+\sum_{j=Q-1}^{R-1}a_{ij}, C_{(i-1)(Q+1)}\right.$
$+S_{(i-1)i(Q+1)}+\sum_{j=Q+1}^{R}t_{ij}+\sum_{j=Q}^{R-1}a_{ij}, \cdots,$
$\left.C_{(i-1)R}+S_{(i-1)iR}+\sum_{j=R}^{R}t_{ij}+\sum_{j=R-1}^{R-1}a_{ij}\right]$

For operation (b), Eq. (12) should be replaced by Eq. (17).

$$C_{ij}^*(pl)=\max[\max\{E_{(i-1)j}, C_{i(j-1)}, E_{(i-1)(j+1)}-t_{ij}-a_{i(j-1)}$$
(for $\max[E_{(i-1)}, C_{i(j-1)}]+a_{i(j-1)} > E_{(i-1)(j+1)}-t_{ij}-a_{i(j-1)}$),
$\min\{\max(SE_1^*(l,k)+as(l,k)),$
$\max(SE_2^*(l,k)+as(l,k)), \cdots, \max(SE_p^*(l,k)+as(l,k)),$
$E_{(i-1)(j+1)}\}-t_{ij}-a_{i(j-1)}]+t_{ij}+a_{i(j-1)}$ (for $j=1, 2, \cdots, Q-1$ and $R+1, R=2, \cdots, M$) (17)

Where $SE_p^*(l,k)$, $SS_p(i,j)$ and $SE_p(i,j)$ have been defined earlier.

A HYBRID SYSTEM OF HEURISTICS AND SIMULATED ANNEALING

In recent years, SA has been successfully used to solve several combinatorial optimization problems. The basic idea behind SA is to consider even a poor solution as an acceptable intermediate solution with a suitable probability for avoiding a local optimum. It uses a concept consisting of different configurations of energies for an analogous physical state. Usually, physical systems, i.e., annealing steel, have many locally minimum analogous states. The algorithm starts with a randomly chosen initial state. In each step of the algorithm, new candidate solutions are generated from the current solution by means of random perturbations. Let E_1 be the objective function value of the current solution and E_2 be that of the new solution. For the case of minimization problems, if $E_2 \leqq E_1$, the new solution becomes E_2 necessarily. But for the case of $E_2 > E_1$, both of E_1 and E_2 can be new solutions with proper probabilities. The probability of accepting E_2 as the new solution is given by $P(\Delta E)=\exp[-(E_2-E_1)/kt]$. A random number which is uniformly distributed in the interval [0, 1] is generated and compared with $P(\Delta E)$. If the generated random number is less than $P(\Delta E)$, the new solution is changed back to the old one. Otherwise, it is discarded and another solution is generated from the current solution. The algorithm continues when a certain termination criterion, such as a prospective number of rearrangements is satisfied. This scheme is known as the Metropolis algorithm.

The efficiency of SA is strongly affected by the following parameters, the initial value of the control parameter T= kt, the number of solutions generated at each T, the decreasing rate of T, and the efficiency in generating new solutions from old solutions. SA is terminated after $3N^3$ (N is the number of products) sequences have been generated. The basic issue is how the algorithm should move from one sequence to another. Ku and Karimi [1991] used a simple strategy of reducing 5 % of kt for each iteration which is composed using 20 discrete iteration steps, i. e., kt remains constant for each step but decreases by 5 % from one group to the next rearrangement with $0.15N^3$ iterations. This means that the final value of kt is 0.95^{20} or approximately 0.36 times the initial kt value.

Thus, the initial probability of up-hill movement suggested by Ku and Karimi [1991] is as follows:

$$P_1(\Delta E)=\exp(-(E_1-E_0)/(kt_1))$$
$$=\exp(-(E_1-E_0)/(1.5(\sum_{i}^{3000}|\Delta E_i|/3000)))\sim 0.513 \quad (18)$$

The last probability is as follows:

$$P_{20}(\Delta E)=0.36\ \exp(-(E_i-E_{i-1})/(kt_i))$$
$$=0.36\ \exp\left(-(E_i-E_{i-1})/(1.5\ (\sum_{i}^{3000}|\Delta E_i|/3000))\right)\sim 0.185 \quad (19)$$

In the case of SA, the initial probability of up-hill movement is nearly 50 percent and the last probability is about 20 percent.

We have developed a new method of SA based on the Metropolis Algorithm called a hybrid system. This system is composed of a two-stage search algorithm. In the first stage, the RAES algorithm was used to obtain a better location of the initial state. RAES has been reported to be a simple and predominant method for flowshop scheduling. RAES was proposed as a method to solve the multiunit UIS scheduling problem. It is a two-phase heuristic with a recursive improvement strategy. In the first phase, a pseudo-two-unit UIS problem is generated from the original M-unit problem as follows:

$$t_{i1}^*=\sum_{j=1}^{M}(M-j+1)t_{ij} \quad t_{i2}^*=\sum_{j=1}^{M}j\ t_{ij} \quad i=1,2, \cdots, N \quad (20)$$

Where t_{ij}^* is the processing time of product i on unit j for the pseudo-two-unit problem. Then Eq. (20) is applied to Johnson's algorithm to generate an initial sequence. In the second phase, the initial sequence is improved by generating (N-1) sequences via pairwise interchanges of adjacent products in the sequence and selecting the best of those as the next solution. The procedure is repeated until no improvement is found. The hybrid system of this study used the result of RAES as the initial sequence.

In small scheduling problems ($N \leqq 10$), average percentage deviations of RAES based on the optimal solutions are reported within 10 percent for a batch flowshop with UIS. In large problems ($N \geqq 15$), they have been shown to be within 20 percent the optimum. For this reason, an initial probability of a hybrid system has not to use $P1(\Delta E)\sim 0.5$ like SA. That's because this value results in giving up a lot of blocks of good solutions. At the second stage, we used a lower probability of up-hill movement.

Thus if it is not a predominant solution, it is hardly accepted as a new one. We guessed and calculated the probability of up-hill movement. We searched for the probability

of up-hill movement between 0.2 and 0.5. So we solved many problems with several initial values of the control parameter T=kt, from 0.6 to 0.95. From the results, we got the optimal initial values of the control parameter T of 0.75 for small size problems and 0.8 for large size problems.

Therefore we controlled initial probability of up-hill movement as follows.

In the case of small size problems:

$$P_1(\Delta E)=\exp(-(E_i-E_{i-1})/(kt_i)) =\exp(-(E_i-E_i-1)/(0.75(\sum_i^{3000}|\Delta E_i|/3000)))\sim 0.264 \quad (21)$$

$$P_{20}(\Delta E)=0.36\exp(-(E_i-E_{i-1})/(kt_i)) =0.36\exp(-(E_i-E_i-1)/(0.75(\sum_i^{3000}|\Delta E_i|/3000)))\sim 0.095 \quad (22)$$

In the case of large size problems:

$$P_1(\Delta E)=\exp(-(E_i-E_{i-1})/(kt_i)) =\exp(-(E_i-E_{i-1})/(0.8(\sum_i^{3000}|\Delta E_i|/3000)))\sim 0.287 \quad (23)$$

$$P_{20}(\Delta E)=0.36\exp(-(E_i-E_{i-1})/(kt_i)) =0.36\exp(-(E_i-E_{i-1})/(0.85(\sum_i^{3000}|\Delta E_i|/3000)))\sim 0.103 \quad (24)$$

The initial probability of up-hill movement is nearly 27 to 29 percent and the last one is approximately 10 percent. The probability of up-hill movement was reduced to around 50 percent from that of SA.

EVALUATION AND CONCLUSIONS

We have tested various problems which were randomly generated to evaluate the performance of this study. The performance measures are makespan and computational time. The size of problems that have been used in this study consisted of comparatively small size problems (N=6, 7, 8, 9, 10) as well as large size problems (N=15, 20, 25, 30, 35). The unit numbers (M) for each problem are 4 and 8. For each problem size, we have tested 20 problems, 10 problems of operation(a) and 10 problems of operation(b), which were generated randomly. In total we have tested 400 problems. For the small size problems, we have assumed one movable vessel and for large ones we assumed two movable storage vessels.

The values of set-up, transfer, and clean-up time were generated randomly between 1 and 9, and the values of processing time were generated randomly between 10 and 99.

We solved these problems using RAES, SA, and the hybrid system for comparing the efficiency of each method.

To analyze the efficiency of each method, we took the RAES makespan as a basis and we defined the superiority(S (%)) as the difference of the result of each method from this basis. The equation for S is as follows:

$$S=\frac{\text{Solution of RAES}-\text{Solution of each method}}{\text{Solution of RAES}}*100 \quad (25)$$

The value of S can be either a positive or a negative number. The method which has the largest value of S is the best method. The actual results are as shown in Fig. 5 and 6. Fig. 5 is the diagram for the small size problems and Fig. 6 is the diagram for large size problems.

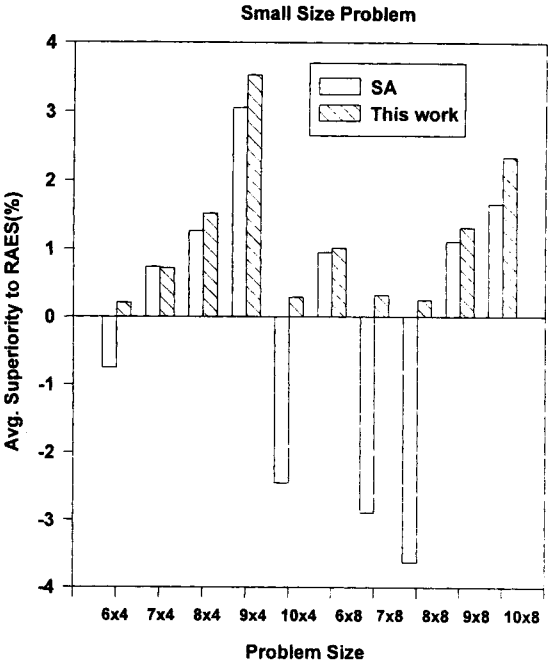

Fig. 5. Superiority to RAES of SA and this work of Small Size Problems.

These figures show that the hybrid system is superior to SA. In the case of the smaller size problems, the results of this system are slightly better than those of SA, but the results of problems like 6×4, 10×4, 7×8 and 8×8 show

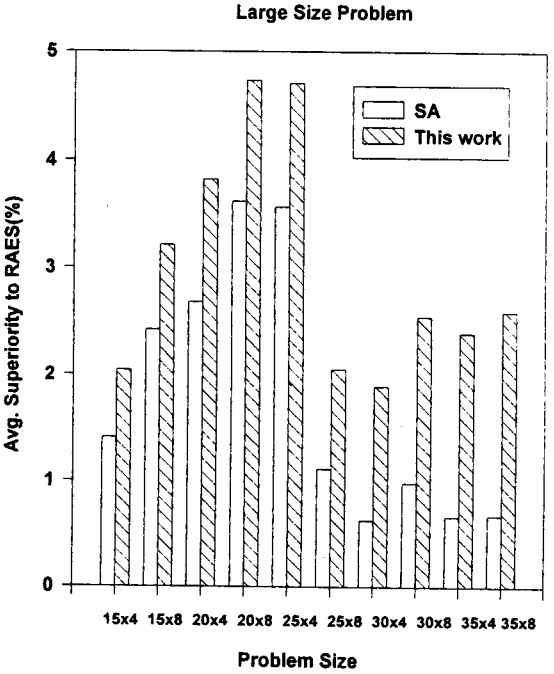

Fig. 6. Superiority to RAES of SA and this work of Large Size Problems.

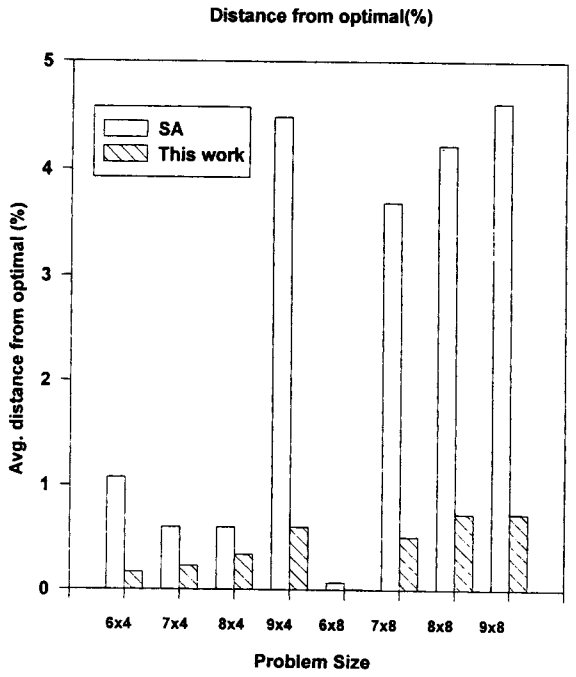

Fig. 7. Average distance from Optimal solution.

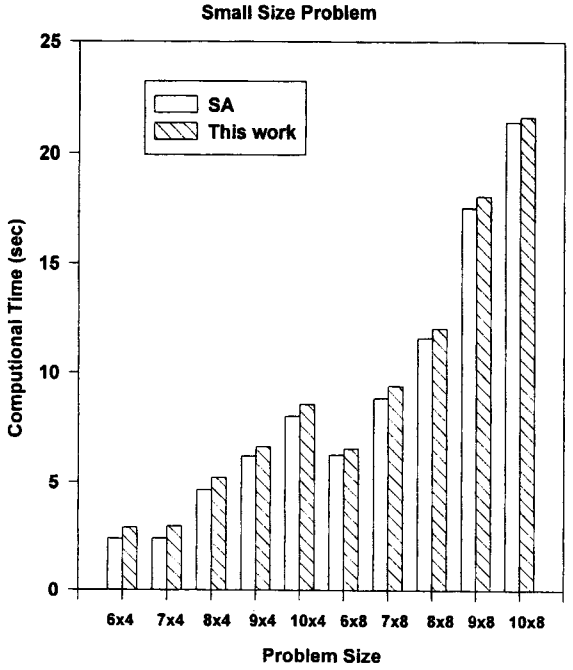

Fig. 8. Computational Time of SA and this work for Small Size Problem.

that superiorities of SA have negative numbers. So compared with this system, there are large differences between the hybrid system and SA. As the size of problems grows, the results become much better than those of SA.

In case of small problems (N=6, 7, 8, 9), we could get the optimal solutions by combinatorial search method. So we also solved the small problems by combinatorial search to check how close between the solution by each method and the optimal solution. We took the optimal makespan as a basis and we defined the optimality (D%) as the difference of the result of each method from the optimal solution. The equation for D is as follows:

$$D = \frac{\text{Solution of each method} - \text{Optimal solution}}{\text{Optimal solution}} * 100 \qquad (26)$$

If the value of Eq. (26) becomes zero, the solution should be optimal. The actual results are as shown in Fig. 7. We find the results of the hybrid system are better than those of SA. Over one third of result sets of the hybrid system are optimal and all of the results by the hybrid system took the values within 1 % distance from optimal.

Next, we measured the computational times of SA and this system. The hybrid system used the results of RAES as an initial sequence as mentioned earlier. The hybrid system had a longer computational time than SA but it reduced the search space required to find the solutions. Therefore, its computational times can be reduced.

The computational time diagrams of SA and the hybrid system are also shown in Fig. 8. and 9.

In the case of the smaller size problems, the computational time of the hybrid system is a little longer, just one second, than that of SA. This means that it takes less time because the search space is not very large and it uses RAES as an initial state. But, as the size of problems grows, the search space increases. In the case of the hybrid system, it proves that the search space is reduced more than that of SA and the computational time is shorter than that of SA.

And we also calculated the computational times of combinatorial search method and this system. The computational time diagrams of combinatorial search method and the hybrid system are shown in Fig. 10. In this figure, we notice that the computational time of combinatorial search method is much

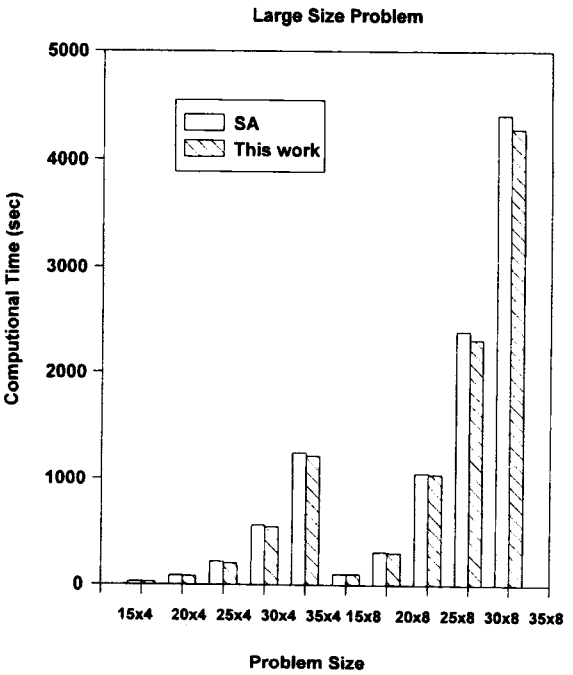

Fig. 9. Computational Time of SA and this work for Large Size Problem.

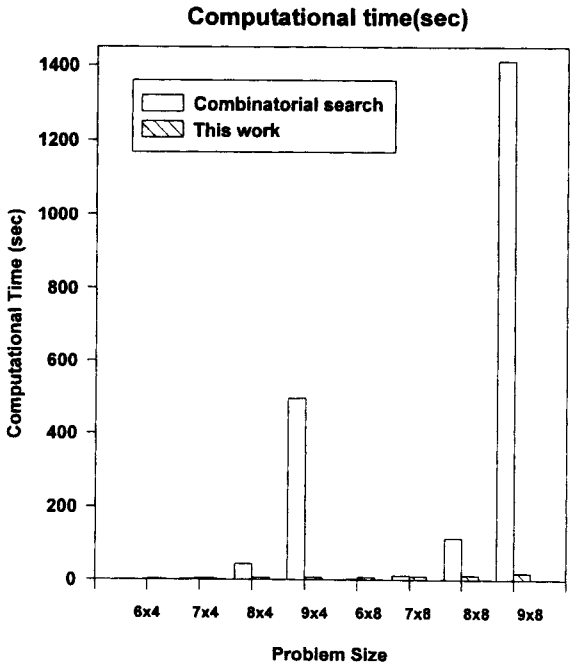

Fig. 10. Comparison of Computational Time of Optimal solution and this work solution.

longer than that of hybrid system as the problem size increases. In the case of problem size, 9×4 and 9×8, the computational times of combinatorial search method are 495 and 1412, respectively. These values are about 80 times of that of hybrid system. It is clear that the combinatorial search method can get an optimal solution. But it takes much longer and it is used for small size problem. This hybrid system can get a nearly optimal solution for small size problem and it takes only a few minutes. And when the size of problem increases, we can get suboptimal solution by using this system.

The computational studies were performed using Pentium 100 MHz and 16 Mbyte RAM under the Windows-95 operating system. The program to evaluate the performance of this study was coded using the C language and was compiled by visual C++.

ACKNOWLEDGEMENT

This research was supported by the Yeungnam University Research Grants of 1997, and the KOSEF through the Automation Research Center at POSTECH. We appreciate the support.

NOMENCLATURE

a_{ij} : transfer time which is required for transferring product i out of unit j to $(j+1)$
$as(l,k)$: storage clean-up and preparing time after using l th product at k unit
C_{ij} : completion time of processing i th product in j th unit
$C_{ij}(pl)$: completion time of processing i th product in j th unit with a pipeless storage
C_{ij}^* : $C_{ij}+a_{ij}$
C_{NM} : completion time of the final product N from unit 1 to the final unit M
E_i : makespan as an energy function
E_{ij} : $C_{ij}+a_{ij}+s_{i(i+1)j}$
k : Boltzman constant
$P(\Delta E)$: probability for up-hill movement
sc : storage checking state
$S_{i(i+1)j}^*$: set-up time which is required for preparing $(i+1)$th product after i th product
$SS(l,k)$: storage starting times for product l after unit k
$SE(l,k)+as(l,k)$: storage ending times for product l after unit k
T_i : temperature as a control parameter
t_{ij} : processing time of product i at unit j

REFERENCES

Hasebe, S. and Hashimoto, I., "Optimal Design a Multi-purpose Pipeless Batch Chemical Plant", Proc. Fourth Intl. Symp. on Proc. Systems Engng, Montebell, Canada, 1991.

Jung, J. H., Lee, H. and Lee, I., "Completion Times Algorithm of Multi-product Batch Processes for CIS Policy with Nonzero Transfer and Set-up Times", *Computers Chem. Engng*, **20**, 845 (1996).

Jung, J. H., Lee, H., Yang, D. R. and Lee, I., "Completion Times and Optimal Scheduling for Serial Multi-product Processes with Transfer and Set-up Times in Zero Wait Policy", *Computer Chem. Engng*, **18**, 537 (1994).

Jung, J. H., Won, K., Jung, J. Y., Yang, D. R. and Lee, I.-B., "Scheduling of Multi-product Batch Processes Using Modified Simulated Annealing", *HWAHAK KONGHAK*, **32**, 753 (1994).

Kim, M., Jung, J. H. and Lee, I., "Intelligent Scheduling and Monitoring for Multi-product Networked Batch Process", *Computers Chem. Engng*, **20**, Suppl., S1149, (1996).

Ku, H. M. and Karimi, I., "An Evaluation of Simulate Annealing for Batch Process Scheduling", *Ind. Engng Chem. Res.*, **30**, 163 (1991).

Ku, H. M. and Karimi, I., "Completion Time Algorithms for Serial Multiproduct Batch Processes with Shared Storage", *Comput. Chem. Engng*, **14**, 49 (1990).

Ku, H. M., Rajagopalan, D. and Karimi, I., "Scheduling in Batch Processes", *Chem. Engng Prog.*, **August**, 35 (1987).

Lee, I., Jung, J. H., Yang, D. R. and Chang, K. S., "Completion Times Algorithm and Scheduling Strategy of Multi-product Batch Processes for Common Intermediate Storage (CIS) Policy: Flexible Intermediate Storage", FOCAPO '93, Crested Butte, U.S.A., 1993.

Niwa, T., "Pipeless Plant Boost Batch Processing", *Chem. Engng*, **June**, 102 (1993).

Niwa, T., "Transferable Vessel-type Multi-purpose Batch Process", Proc. Fourth Intl. Symp. on Proc. Systems Engng, Montebello, Canada, 1991.

Reklaitis, G. V., "Perspectives on Scheduling and Planning of Process Operations", Proc. of Fourth International Symp. on PSE, Montebello, Canada, 1991.

Reklaitis, G. V., "Review of Scheduling of Process Operations", AIChE Symp. Ser., **78**, 119 (1982).

Shimatani, T. and Okuda, O., "Pipeless Batch Chemical Plants Offer a New Approach", *Chem. Engng*, **October**, 181 (1992).