NEW OPTIMIZATION APPROACH FOR SCHEDULING THE BATCH HEAT TREATMENT PROCESS WITH SEQUENCE DEPENDENT SETUP TIMES

Ehab A. Abdelhafiez *, Tawfeeq Alkanhal **

* Mechanical and Industrial Engineering Department, Faculty of Engineering, Majmaah University, Saudi Arabia (e-mail: ehabaty@yahoo.com).
** Computer Engineering Department, Faculty of Engineering, Majmaah University, Saudi Arabia (e-mail: t.alkanhal@mu.edu.sa)

ABSTRACT
The Shaking Optimization Algorithm is among the newly developed optimization algorithms that belong to the Evolutionary Computations Algorithms. It combines heuristics with randomness in the evolution process rather than just apply operators e.g., mutation, recombination, and/or selection rules. It has been used successfully to solve the classic Job Shop Scheduling Problem, the Cutting Stock Problem, and the Unconstrained Engineering Optimization Problems, and the results show that the algorithm is a good competitive to other evolutionary computation algorithms e.g., the Genetic Algorithm and the Particle Swarm Optimization regarding solution time, efficiency, and complicity. This paper uses the SOA algorithm to solve the scheduling problem of the Batch Annealing Process with Sequence-Dependent Setup Times in the heat treatment of steel coils as one of complex scheduling problems. The objective is to determine the movement schedule of shared equipment among parallel bases that gives minimum makespan for processing a given batch size. The bell-type batch annealing process is used here for the comparative study. The results show that the proposed approach is able to solve this class of problems as well in a reasonable time and gives competitive results to other algorithms.

KEY WORDS
Shaking Optimization Algorithm (SOA); Evolutionary Computations; Job Shop Scheduling problem; Batch Annealing Process

1. INTRODUCTION

Many optimization algorithms have been proposed for finding optimal/ near-optimal solutions to engineering problems for which analytical methods do not apply. The Evolutionary Computations Algorithms (ECs) are based on the processes of evolution in nature. They emulate the evolution of a population of structures based on the measured performance of each member of this population. Emulated evolution is implemented by the application of mutation, recombination, and/or selection rules to determine which members of the population survive and reproduce. With each subsequent generation, the population most likely shows improvement in measured performance, until some terminating condition is met [1].

Many of the Evolutionary Computations Algorithms showed some types of limitations as they are not suitable for fine tuning structures. These limitations are because that these algorithms neglect conventional heuristics, where, in many optimization problems, the optimal solution can be obtained using some heuristics as in the one-machine Job Shop Scheduling problem where the optimal solution is found using the shortest processing time rule, while in two-machine Job Shop Scheduling problem, the solution is found using Johnson heuristic. In general, the hybrid approach often outperforms either method operating alone as indicated by Moraglio et al [2]. Hence, it becomes a common practice to improve the performance of these ECs algorithms by incorporating different search and heuristic techniques. Accordingly, the basic genetic algorithm is hybridized with a Tabu search optimization as done by Moraglio et al [2], simple priority rules is incorporated with the GA as in the work of Hasan et al [3]. Also, Tasgetiren et al [4] hybridized the PSO algorithms with an efficient local search method based on a variable neighborhood search (VNS) method in order to improve the solution quality.

Recently, in 2011, E. Abdelhafiez and F. Alturki [5] introduced the Shaking Optimization Algorithm (SOA) to emulate the natural shaking process that allows items to change their positions due to the differences between their weights/volumes/shapes...etc. The SOA follows the common methodology of the Evolutionary Computations algorithms. This SOA algorithm has been applied to solve the classical Job Shop Scheduling Problem (JSSP) [5, 6, and 7], the “Two-Dimensional Irregular Strip-Packing Problem” [8], and the Unconstrained Engineering Optimization problems [9], the results show that this SOA algorithm outperforms the classical Genetic Algorithm “GA”, Particle Swarm Optimization “PSO”, Simulated Annealing “SA”, and Tabu Search “TS” in solving these classes of problems. In addition, the SOA algorithm outperforms the Genetic Algorithm hybridized with Simulated Annealing Algorithm, while it gives the same results exactly as does the Genetic Algorithm hybridized with two exchange local searches in solving the same classes of problems.
In this paper, the scheduling problem of the Batch Annealing Process with Sequence-Dependent Setup Times in the heat treatment of steel coils is selected to test the capability of the Shaking Optimization Algorithm to solve more complicated problems than the ones already tested.

The batch annealing process consists of a number of bases, number of furnaces, number of coolers, number of coil stacks, and one crane. Both the number of furnaces and the number of coolers are less than the number of bases. Thus, the cranes, furnaces and coolers shared between the bases and considered as limited resources. In addition, a different crane tool is required for moving coils or equipment. Therefore, the time to change these tools is classified as a sequence-dependent setup time. Covers and convector plates are of sufficient numbers so they are not limited resources.

Some research concerning the scheduling problem of the Batch Annealing Process with Sequence-Dependent Setup Times in the heat treatment of steel coils were reported e.g., Moon and Hrymak [10], Dong et al. [11], Quan-li Liu et al. [12], Lixin Tanga et al. [13], and Miranda et al. [14].

Moon and Hrymak [10] proposed a Mixed-Integer Linear Programming (MILP) model. They use the branch and bound method to solve the model, and in order to obtain a feasible solution with short computation time, they divide the equipment scheduling into two groups. Group 1 contains only a crane, and group 2 contains all equipment except the crane. And to reduce the number of variables and constraints, they select a suitable path for time slots through a preordering method. Dong et al. [11] reduces the 10 stages in an annealing cycle to only heating and cooling stages. The problem is simplified as a mixed flow shop sorting problem to be solved by a greedy sorting algorithm. Liu et al. [12] handles a more difficult problem that is more close to real industrial application. They developed a scheduling method by combining an improved genetic algorithm and a condition-based discrete event simulation approach. Lixin Tanga et al. [13] formulated a Mixed Integer Linear Programming model (MILP) through which they show that the problem is NP-hard in the strong sense. Hence, they constructed a two-phase algorithm for the problem. In the first phase, a fully polynomial time approximation scheme is developed for the assignment sub-problem. In the second phase, a heuristics is proposed for the scheduling sub-problem.

This paper starts by a description for the batch annealing process, followed by the definition of the problem. Then, an overview of the Shaking Optimization Algorithm “SOA” is presented followed by a section for the application of this SOA algorithm to solve the batch annealing problem. At the end, comes the Comparative Study that analyzes the effectiveness and efficiency of the proposed approach in solving the concerned problem relative to some other solution approaches.

2. BATCH ANNEALING PROCESS

The batch annealing process consists of a base, a coil stacks “charge”, an inner cover, a furnace, a cooler, and a crane. The base is fixed on its place then; coil stacks, an inner cover which fits over the stacks, a furnace/ or a cooler are placed upon the base in sequence. The movement of each part is done with an overhead crane. Figure 1a shows a general processing flow for the process, while figure 1b shows a schematic of the process cycle.

Figure 1: (a) Processing flow of batch annealing processes, (b) General cycle of batch annealing operations [10].

According to the general cycle of the batch annealing operation shown in figure 1(b), there are mainly 10 stages in the cycle as described by Moon and Hrymak [10] that are:

- Stage 1: Load stack of coils and convector plates upon a base,
- Stage 2: Load a cover,
- Stage 3: Load a furnace,
- Stage 4: Heat the coil stack,
• Stage 5: Unload the furnace,
• Stage 6: Load a cooler,
• Stage 7: Cool the coil stack,
• Stage 8: Unload the cooler,
• Stage 9: Unload the cover, and
• Stage 10: Unload the coils and the plates.

In addition, a deoxidizing gas to purge the air from the space under the cover is to be used just after the cover is loaded. Also, after the furnace is removed, a natural-cooling step may be used before the cooler is placed to decrease utility cost. Although the addition of gas and natural cooling steps are involved in the processing recipe, these steps were not included in the processing stages because none of the process-equipments is used.

Moreover, there are many crane-tools; three types are considered herein this paper: coil-tongs to move a steel coil, hood-lifter to move a steel cover, and magnet to move a convector plate. The suitable tool must be installed as required based on the equipment to be moved. Therefore, the time necessary for the crane-tool change is considered as a setup time. Usually, the crane has two hooks, small and large. The small hook is used to install a magnet and to move a plate. The large hook is used to move a furnace and cooler, to install coil-tongs for moving a coil, or to install a hood-lifter for moving a cover. All stages, except stage 4 (heating) and stage 7 (forced cooling), are loading- or unloading- stages which require a crane. The furnace or the cooler cannot be used for heating or cooling the base while being moved by a crane (stages 3, 5, 6, and 8).

3. PROBLEM DEFINITION

The classical Job Shop Scheduling Problem (JSSP) can be described as follows [15]: we are given a set of jobs and a set of machines. Each job requires a number of operations; each operation needs to be processed for a specific time period on a given machine. Each machine can handle only one job at a time. The purpose is to find the schedule that gives the minimum finish time for all the given jobs. The short term scheduling of the batch annealing process described above is classified as a complex scheduling problem because it has many constraints regarding setup times, transfer times, and intermediate steps that consumes time without using any of the limited resources. The setup and transfer times are considered before the stage begins, while the intermediate steps (gas addition and natural cooling steps) are considered after the stage ends. These in addition to some restricted operation-sequence where, once a furnace is loaded, the heating must start directly without any delay. And the same restriction exists for the cooler. Also, to place the furnace/ or the cooler upon a base (stage 3 or stage 6 for this base), it is necessary to call the corresponding unloading of this furnace / or cooler (stage 5 or stage 8) for some other base.

In this problem, the shared equipments are considered as fixed machines while the bases as movable jobs. The problem can be stated as:

Determine the movement schedule, of shared equipment among parallel bases, that gives minimum makespan "finish time" for processing a given batch size.

Given:
• A set of bases, represent the jobs required,
• A number of processing stages, represent the processing operations required for each job,
• A set of process equipment units, represent the machines available.
• Processing time for each stage for each base,
• Setup requirement for each stage (if exists),
• Setup and transfer times, and
• Times for the gas-addition and for the natural-cooling step for each base.

The mathematical modeling of the problem involves a very large number of integer variables and constraints. In the MILP model presented by Moon and Hrymak, and for a simple case of two bases, two furnaces, two coolers, and one crane there have been 1030 constraints, 154 integer variables, and 41 continuous variables. For larger problem sizes, this MILP failed to obtain a feasible solution due to the nature of the branch and bound method. Therefore, to find a feasible scheduling solution for the problem with a larger number of bases, Moon and Hrymak divide the equipment scheduling into two groups. Group 1, contains only a crane, and group 2, contains all equipment except the crane and hence, the obtained solutions are not guaranteed to be optimal.

Liu et al [12] mentioned that, if all the constraints are considered, it will be very hard to formulate a MILP model. Even if the MILP model is formulated, when the number of bases, furnaces or coolers goes up, the size of the MILP model will increase correspondingly, making it difficult to solve. Also, Lixin Tanga et al. [13] formulated a mixed integer linear programming model to solve the problem. They mentioned that a computational experiment attempting to solve a model with only eight jobs using CPLEX on a PC showed that the preprocessing alone already takes over 40 min (considering the CPU speed at 2009). Moreover, they mentioned that, it is not possible to find a good feasible solution in a practical amount of time. This motivates many researchers to exploit the problem properties for approximation-algorithms using evolutionary computations.

4. OVERVIEW OF THE SHAKING OPTIMIZATION ALGORITHM

Shaking Optimization Algorithm “SOA” has been proposed in a trial to overcome some of the limitations exist by other evolutionary computational algorithms. It follows the common methodology of the Evolutionary Computations. It starts with an initial random solution,
then it evolutes to generate more efficient solutions. The evolution is implemented using a number of heuristics, it comprises three main processes; organizing items in some order, closing gaps that may exist between these items while changing their positions, and swap some pairs of items in a fine tuning process.

The SOA consists of four stages; namely, Local Search, Collision, Fine Tuning and Global Optimization. In the local search, based on the shaking strength, a number of items change their positions to reorganize themselves in some order e.g., in ascending order according to some criteria called “weight”. So, these changes in position are a result of the inertia of items such that the items with higher weight go far away and so on. The shaking process in the local search stage may result in some gaps in-between items. These gaps cause collision between items. The definition of “gap” depends on the problem under consideration. In the JSSP, the gap may be the delay time of a job or be the idle time of a machine, where in the Cutting and Packing problem, it may be the difference between the number of entities or be the difference between areas... etc.

To identify gaps, after each shake, a scanning process is carried out over the arrangement to define a gap-index for each item. The gap-index takes a value from 0 to 3. During these arrangements of items, some form of fine tuning will happen to improve the orientation of items for better fit in place.

In Fine Tuning stage, some items with gaps around start to swap or rotate resulting in better penetration. The items allowed to swap or rotate are those items come just after a gap or followed by a gap. The rule that best describes the swapping action is the one of Nowicki and Smutniciciki (1996) and has been described by M. Emin and Terence C. (2004). The rotation of items is applicable in Cutting Stock problems as the orientation of items affects greatly the solution efficiency.

After a certain number of iterations “Shakes”, or at any other stopping criteria, a new initial solution is to be developed based on the best solution so far. The approaches that have been used in solving the JSSP [5, 6, and 7] and the “Two-Dimensional Irregular Strip-Packing Problem” [8] in regards to this stage is to recover from current local search by developing new solution through keeping some blocks from the current solution while changing their positions randomly. The blocks that will remain are the compact blocks have no gaps in-between. So, although this will generate new solution with less efficiency than the original one, it guarantees that the chance for this new solution to evolve in a converging manner is high. In solving the engineering optimization problems [9] that are mathematical formulas, this new initial solution “searching pattern” is to be developed by using the best values so far of the decision variables as the starting point “seed” of the new search pattern to not search from the minimum point every new-iteration.

5. APPLICATION OF THE SOA TO SOLVE THE BATCH ANNEALING PROCESS PROBLEM

The proposed solution approach comprises two modules: module one uses the SOA algorithm to generate the movement plan (sequence) of the shared equipments among the bases for processing the given batch size. Module two simulates the batch annealing process; through applying the movement plan generated by the first module; to calculate the process makespan and to define gaps (delays), then it sends a feedback to the SOA in form of a gap-matrix for fine-tuning the movement-plan.

5.1 Module 1: Generation of Movement Plan for the Shared Equipments

The generation of the movement plan for shared equipment is performed by applying the SOA algorithm where the problem is to be represented using a suitable coding system, then the SOA algorithms proceeds.

5.1.1 Proposed Coding System

According to Mehmet and M. Emin (2006), schedules are represented using a set of integers; each integer stands for an operation, while these integers do not represent certain operations. This way of representation prevents infeasibility, and always provides a feasible active schedule. This means that, each job is represented m-times within the chromosome. For a 3-Jobs/ 3-Machines problem, the code of [2 1 2 2 1 3 1 3 3] means Job-2/ Operation-1 followed by Job-1/ Operation-1 and so on, i.e., this code means [o_{21}, o_{11}, o_{22}, o_{23}, o_{13}, o_{13}, o_{13}, o_{33}] where o_{ij} stands for the j^{th} operation of i^{th} job.

For the SOA algorithm, another digit must be used that refers to the Gap-index of the item. Hence, the proposed coding system consists of a set of integers, each is of two parts e.g., [2,1/1,0/2,1/2,3/,1,0/3,1/1,2/3,0/3,0].

5.1.2 Definition of the SAO Parameters

A great attention has been paid in finding the best setting of the SOA algorithm in solving the intended problem. The best setting of the algorithm parameters is as follows:

- Number of dimensions = 6,
- Weight criteria is processing time,
- Shaking force = random number between 2 and 6.
- Number of generations = S. force × (n + m), where: (n) number of bases, and (m) number of stages
- Number of shakes per generation = (S. force × n × m).
5.1.3 Evolution of Solutions

A number of random solutions (around 10) are to be developed and tested and the best one is selected to be the initial solution of the SOA algorithm. An array of a number of random elements based on the shaking force is to be developed. The elements of this array are the elements that are going to exchange their positions during the shaking process. By shaking, each of the priority rules; that represent the movement along one of the directions (coordinates); is to be applied individually, and if improvement results, the resulting solution will become the new initial solution, and the remaining rules are to be applied considering this new solution. After investigating each of the six-directions (coordinates), the developed solution (sequence/schedule) is to be evaluated by calculating the resulting makespan (using the second module). During this evaluation process, the delay time of each operation is calculated, and hence the gap index is defined. Based on this gap-index, the collision process described above is performed and the solution is to be evaluated again after each collision and the gap index is to be updated. By the end of this collision process, and based on the updated gap-index, the fine tuning process is to be carried out using the method of Nowicki and Smutnicki (1996). The shaking, collision, and fine tuning processes are to be repeated for a certain number of iterations “Shakes”, or to any other stopping criteria. Following this, a new initial solution is to be developed based on the best solution so far by keeping some blocks from the current solution while changing their positions randomly. The blocks that will remain are the compact blocks having no gaps in-between.

5.2 Module 2: Simulation of the Batch Annealing Process (Calculation of makespan)

The output of the SOA is a sequence of the whole operations (stages) required for the processing of all the bases. This module simulates the annealing process following this sequence and sends a feedback to the SOA in form of the gap-index. The simulation runs as follows:

The first element of the given sequence refers to one of the given bases, and because this is the first existence of this base in the sequence, it is for the first stage. The setup matrix is to be investigated to check if setup time should be considered for this stage. Then, the finish time of this base and the equipment used are to be defined based on the related stage processing time and setup time needed. After that, the second element of the sequence is to be picked up. If it is for the same base, then it will be for the second stage, and hence the process of setup needs and finish time estimation is to be repeated. While, if this second element refers to another base, then it will be for the first stage of this base. And again, the process of setup needs and finish time estimation is to be repeated and so on.

In calculating the finish time of any stage, two cases may exist; the previous finish time of the required equipment is greater than that of the considered base, or vice-versa. Then, the new finish time for both of them will equal the latest finish time (among them) plus the new stage processing time in addition to any setup time that may exist.

And, in considering the restriction of starting heating/or cooling once the furnace is loaded/or the cooler is unloaded respectively, the previous finish time of the carne, the concerned base, and the furnace/or cooler must be considered simultaneously. Then, the new finish time for the carne will equal the latest finish time (among all of them) plus the loading/or unloading processing time in addition to any setup time that may exist, while the finish time of the concerned base and the furnace/or the cooler will equal the new finish time of the carne plus the heating/or cooling time respectively, in addition to any setup time that may exist.

Once a difference between the previous finish-time of the concerned base and its new starting time exists, a delay exists that equals to this time difference. To consider the time for the gas-addition or for the natural-cooling steps, after calculating the finish time of stage two (loading a cover), the time needed for the gas-addition is to be added to the finish time of the concerned base. Similarly, after calculating the finish time of stage five (unloading the furnace), the time needed for the natural-cooling step is to be added to the finish time of the concerned base.

6. COMPARATIVE STUDY

In order to evaluate the efficiency and effectiveness of the proposed solution approach, a comparative study needs to be carried out. However, the only data available in literature relating to the concerned problem is that of Moon and Hrymak [10]. Table 1 lists the processing times data for 12 bases of the batch annealing problem as presented by Moon and Hrymak.

<table>
<thead>
<tr>
<th>Base</th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
<th>Stage 4</th>
<th>Stage 5</th>
<th>Stage 6</th>
<th>Stage 7</th>
<th>Stage 8</th>
<th>Stage 9</th>
<th>Stage 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>B</td>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.6</td>
</tr>
<tr>
<td>C</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>D</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>E</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>F</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>G</td>
<td>0.6</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.6</td>
</tr>
<tr>
<td>H</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>I</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>J</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>K</td>
<td>0.8</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.8</td>
</tr>
<tr>
<td>L</td>
<td>0.4</td>
<td>0.3</td>
<td>0.5</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.3</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Table 1. Processing-times of the batch annealing problem [10].
Table 2 lists the times required for gas addition and natural cooling step according to Moon and Hrymak. Also, the setup times $S_{ij}$ that depends on the two-movement stages $i$ and $j$ of the crane are listed in table 3. All stages, except stage 4 (heating) and stage 7 (forced cooling), are loading- or unloading- stages which require a crane. The furnace or the cooler cannot be used for heating or cooling the base while being moved by a crane (stages 3, 5, 6, and 8).

Table 2. Times required for gas addition and natural cooling.

<table>
<thead>
<tr>
<th>Base</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gas addition</td>
<td>0.3</td>
<td>0.6</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>Natural cooling</td>
<td>5</td>
<td>25</td>
<td>40</td>
<td>15</td>
<td>26</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Base</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gas addition</td>
<td>0.3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>Natural cooling</td>
<td>9</td>
<td>14</td>
<td>22</td>
<td>35</td>
<td>24</td>
<td>11</td>
</tr>
</tbody>
</table>

A total of 50 runs are performed to solve each of the problem-cases shown in table 4. The results obtained using the proposed solution approach vs. that of MILP-2 by Moon and Hrymak are shown in table 3.

The MILP-2 model evolves dividing the equipment scheduling into two groups. Group 1, contains only a crane, and group 2, contains all equipment except the crane. Also, a suitable path for time slots is selected to reduce the number of variables and constraints. The table lists the makespan, the percentage improvement, the average number of total shakes (including number of generations) to get the best solution, and the average time (in seconds) to obtain this best solution.

The results show that the proposed approach outperforms the MILP-2 in solving all the concerned cases of the problem while the improvement ranges from 0.29% to 7.46%. Also, both algorithms give the same makespan for the 4-base and 5-base cases (3F- 3C), while the average solution time of the second case is little bit larger than that of the first case. The similarity of the makespan may be because the bottleneck base is one of the first four bases which justify the little bit difference in solution times. This results indicates the consistently of the algorithm.

The solution results of the SOA algorithm in solving the 10 stages, 12 bases for the case of 2 furnaces, 2 coolers, and one crane using 80 generations and 240 shakes is shown in figure 2. This figure shows the start and finish times for each stage of each base, where $c$ refers to crane, $H1$ refers to heater number 1, $Cl1$ refers to heater number 1 and so on.

The comparative study of the proposed approach with the Mixed-Integer Linear Programming MILP-1 and MILP-2 of Moon and Hrymak, shows that the proposed approach is simple to apply, effective as it give better results for all the indicated problem-cases, in addition to its relative speed where, all solutions are obtained in a time range between 0.5 to 134 seconds in comparison with the 40 minutes time needed to solve the eight-jobs problem (considering the CPU speed at 2009) as indicated by Lixin Tanga et al. (2009).

Regarding the work of; Dong et al. (2001), Liu et al. (2005) and Lixin Tanga et al. (2009), there is a lack of numerical data published for the cases they have been studied. Also, none of them have hold any comparison or made any comments regarding the ability of their proposed solution algorithms to solve the cases of Moon and Hrymak (1999) although all of them consider the basics and definition of the same problem of Moon and Hrymak. Dong et al. [11] simplified the problem to become a mixed flow shop sorting problem to be solved by a greedy sorting algorithm. Liu et al. [12] handles a more difficult problem that is more close to real industrial application. However they used the data obtained from the real-time database of the supervisory control computer in the annealing shop for the analysis of their proposed method in comparison with the results of the worker’s scheduling. Lixin Tanga et al. [13] formulated a mixed integer linear programming model to solve the problem.
They mentioned that a computational experiment attempting to solve a model with only eight jobs using CPLEX on a PC showed that the preprocessing alone already takes over 40 min (considering the CPU speed at 2009). They used randomly generated problem instances to analyze the performance of their proposed algorithm. Hence, and due to the characteristics of the proposed solution algorithm specially the handling of the gap-closing through the interaction between the two modules, in addition to the solution time shown in table 4 above, it is expected (theoretically) that this approach is able to give the same/ or even better results compared with other solution algorithms as that of Dong et al., Liu et al., and Lixin Tanga et al.

Figure 2: The solution results of the SOA algorithm in solving the 10 stages, 12 bases batch annealing process with 2 furnaces, 2 coolers and one crane with setup times using 80 generation and 240 shakes.

7. CONCLUSION

In this paper, a two-module solution approach is proposed for the scheduling of the Batch Heat Treatment Process with Sequence Dependent Setup Times. Module one applies the Shaking Optimization Algorithm “SOA” to generate the movement plan (sequence) of the shared equipments among the bases for processing the given batch size. This SOA algorithm follows the common methodology of the Evolutionary Computations where evolution is implemented by the application of various heuristic rules. Module two simulates the batch annealing process; through applying the movement plan generated by the first module; to find the process makespan and to define the areas of improvement (gap areas), then it sends a feedback to the SOA in form of a gap-matrix for the fine-tuning of the movement-plan. The bell-type batch annealing process of Moon and Hrymak is concerned herein to evaluate the performance of the proposed algorithm vs. the Mixed-Integer Linear Programming (MILP) solutions. The results of the comparative study show that the proposed approach outperforms the 2-group mixed-integer linear programming (MILP-2) model of Moon and Hrymak in all cases including the 2 furnaces, 2 coolers and the 3 furnaces, 3 coolers with respect to both the solution efficiency and speed. In addition, it uses shorter time than that of Lixin Tanga et al. Also, and due to the interaction between the two modules of the proposed solution that makes the solution approach able to work with specific portion of the solution (schedule plan) for improvement, the approach expected to give same /or better results of other solution algorithms e.g., that of Dong et al. (2001), Liu et al. (2005) and Lixin Tanga et al. (2009).
REFERENCES


