doi: 10.2306/scienceasia1513-1874.2005.31.383

Energy-Based Workforce Scheduling Problem: Mathematical Model and Solution Algorithms

Kriangchai Yaoyuenyong[a] and Suebsak Nanthavanij[b*]

[a] Industrial Engineering Program, Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12121, Thailand.
[b] Management Technology Program, Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12121, Thailand.

* Corresponding author, E-mail: suebsak@siit.tu.ac.th

Received 16 Feb 2005
Accepted 6 Jul 2005

ABSTRACT: A workforce scheduling problem to schedule the minimum number of workers to perform a set of physical tasks such that their daily energy capacities are not exceeded is discussed. Firstly, a mathematical model is formulated. Two heuristics and one exact algorithm are then explained. Based on our computational experiment, a hybrid procedure consisting of the above mentioned algorithms is found to be very efficient in solving this workforce scheduling problem to optimality.

KEYWORDS: combinatorial optimization, workforce scheduling, bin packing.

INTRODUCTION

Workforce scheduling is indispensable in most complex production and service systems such as factories, constructions, airports, banks, and hospitals. It enables the system to implement multiple tasks within specified work duration under available resources and, in particular, limited workforce. Workforce scheduling problems using combinatorial methods have been continuously studied and well known for their applications. However, most previous works concern only the completeness of tasks and scheduling cost.[1-4] In this paper, we deal with the workforce scheduling problem which emphasizes the health and safety issue of workers.

Industrial workers are often required to expend moderate to high level of physical energy to perform their assigned tasks such as lifting/lowering, carrying, loading/unloading, and operating production machines. If the workloads are too strenuous, workers are likely to develop excessive muscular and whole-body fatigue, which can cause occupational accidents and over-exhaustion after work. Workforce scheduling, commonly known as job rotation, is frequently recommended as a means to reduce the risk of over-exhausion.[5] When assigning a worker to perform a physical task, it is necessary to know the amount of energy required to perform the task (or energy cost) and the energy capacity of the worker (or energy limit/budget). Based on an ergonomic recommendation, the daily (8-hr workday) energy expenditure of a worker should not exceed 33% of his/her energy capacity.[5] (For simplicity, we shall refer to this recommended capacity as the *working* energy capacity.) For different individuals, their working energy capacities normally vary. Depending on the energy costs of required tasks, the number of workers who can be safely assigned to perform the tasks on a rotational basis might have to be greater than the number of tasks. The energy-based workforce scheduling problem (WSP-E) is thus intended to find the minimum number of workers and their daily work assignments such that their working energy capacities are not exceeded. To some extent, WSP-E has some features that are similar to those in the scheduling problem with discrete *nonrenewable* resources.[6] Nevertheless, to our knowledge, a detailed procedure for WSP-E has never been discussed in the literature.

The remainder of this paper is organized as follows. Firstly, we introduce a mathematical model of WSP-E. Then, we propose efficient algorithms for finding the optimal workforce and their safety daily work assignments. After that, a hybrid procedure for solving WSP-E is discussed. A numerical example of WSP-E is given and solved by the proposed algorithms and the hybrid procedure. Finally, a computational experiment is described and its results discussed.

MATHEMATICAL MODEL OF THE ENERGY-BASED WORKFORCE SCHEDULING PROBLEM

Our strive to study the WSP-E is inspired by the fact that industrial workers are everyday at risk of occupational injuries and illnesses even when performing simple physical tasks such as lifting and carrying. The main causes are over-exertion and over-exhaustion. When the injury or illness is occupation-related, its consequence is more severe than just the worker's medical expense. Only in U.S.A., the Department of Labor's Annual Survey of Occupational Injuries and Illnesses (ASOII) states that conservative estimates of direct cost, based on workers' compensation payments (indemnity and medical services) and other direct costs, are at least ten billion dollars per year.[7] The total cost to society is believed to be substantially higher due to various indirect costs, e.g., lost productivity, costs of hiring and training replacement workers, overtime, administrative costs, and miscellaneous transfer payments.[8] As a result, it is greatly important to improve the workplace safety.

One of the most frequently recommended preventive approaches to reduce the worker exposure to hazardous tasks is *job rotation*.[5] When implementing job rotation, it is necessary to assign workers to alternately perform various tasks in different work periods during an 8-hour workday. In this way, the impact of hazardous tasks can be split and shared by many workers, instead of concentrating on some particular workers. The determination of worker-task-period assignments is also known as *workforce scheduling*. A recent study conducted in the Philippines showed that workforce scheduling helps to reduce the risk of occupational hazards in material handling.[9] Workforce scheduling not only significantly enhances the safety of workers but also increases their productivity by sharing energy-demanding tasks among workers.[9, 10]

The WSP-E is intended to determine the worker-task-period assignments for m workers and n tasks such that the number of workers is minimized and, for each worker, the total energy cost does not exceed the daily working energy capacity. It is essential to determine the workers' work schedules for an 8-hour workday to correspond to the ergonomically recommended daily working energy capacity. As commonly practiced in job rotation, a workday is divided into p equal work periods and workers are rotated only at the end of the work period. Since the length of work period is constant for practical management, WSP-E is a discrete scheduling problem. Unfortunately, efficient algorithms for determining an optimal set of worker-task-period assignments for WSP-E have never been developed.

WSP-E can be viewed as a variation of the classical bin packing problem (BPP), an *NP*-hard problem.[11] Researchers are still studying BPP extensively since it plays an important role as the sub-problem of many real-world applications. As a result, new efficient BPP algorithms have been developed recently.[12-14] An excellent survey of existing BPP algorithms is discussed in Coffman et al.[15] Stated completely, BPP is intended to find the minimum number of equal-sized bins that is sufficient for being packed by a set of items with various sizes. An item in BPP is analogous to a physical task in WSP-E, whereas a bin is analogous to a worker. Since workers have different working energy capacities, WSP-E can be viewed as the variable-sized bin packing problem (VSBPP), which is BPP with an additional assumption that bins have different sizes.[16] New algorithms for VSBPP can also be found in the literature.[17, 18] Assumptions required for the formulation of WSP-E are as follows.

1. For a set of n physical tasks being considered, their energy costs are known.
2. All tasks are to be performed eight hours per day.
3. The number of available workers m is known and is more than the number of tasks. Not all workers need to be chosen for inclusion in the workforce scheduling and job rotation. With respect to work efficiency, all m workers are identical.
4. The daily working energy capacities of the m workers are known and are unequal.
5. A workday is divided into p equal work periods.
6. In each work period, a worker can be assigned to only one task and a task needs only one worker to perform.
7. A worker does not have to work in every work period.

The notation used in WSP-E is as shown below.

e_j	energy cost per work period of task j
E_i	daily working energy capacity of worker i
m	number of available workers
n	number of physical tasks
p	number of work periods per day
x_{ijk}	1 if worker i is assigned to task j during work period k; 0 otherwise
y_i	1 if worker i is chosen from the workforce to perform any task; 0 otherwise

WSP-E can be mathematically expressed as follows.

$$\text{Minimize } \sum_{i=1}^{m} y_i$$

$$\text{subject to } \sum_{j=1}^{n}\sum_{k=1}^{p} e_j x_{ijk} \le E_i y_i \quad \text{for } i = 1,\ldots, m$$

$$\sum_{j=1}^{n} x_{ijk} \le 1 \quad \text{for } i = 1,\ldots, m;\ k = 1,\ldots, p$$

$$\sum_{i=1}^{m} x_{ijk} = 1 \quad \text{for } j = 1,\ldots, n;\ k = 1,\ldots, p$$

$$x_{ijk} = \{0,1\} \quad \text{for } i = 1,\ldots, m; j = 1,\ldots, n; k = 1,\ldots p$$

$$y_i = \{0,1\} \quad \text{for } i = 1,\ldots, m$$

While items in BPP can be freely packed into a bin, worker-task-period assignments in WSP-E must not violate the work period assumption (Assumption No. 6). With its complex decision variables and additional constraints, WSP-E is at least as difficult as BPP. It is thus essential to develop an efficient solution procedure to determine either a *near-optimal* or *optimal* solution for WSP-E.

Proposed algorithms for WSP-E

In this section, a method for identifying a lower bound of WSP-E is firstly given. Then, two approximation algorithms and an exact algorithm for solving WSP-E are proposed.

Lower Bound

Letting $n \le d \le m$ and $E_1 \ge E_2 \ge \ldots \ge E_m$, a lower bound (LB) of WSP-E is defined as follows.

$$\text{LB} = \min\left\{ d \,\middle|\, \sum_{i=1}^{d} E_i \ge p \sum_{j=1}^{n} e_j \right\} \tag{1}$$

M²-FFD Heuristic

The M^2-FFD heuristic is modified from the well-known BPP heuristic, called "First Fit Decreasing."[19] M^2-FFD is implemented as follows. Let I be a set of energy costs consisting of p copies of each energy cost e_j of all tasks; sort them in non-increasing order. Given a set of m workers, arrange them in non-increasing order of the working energy capacity. Next, index the workers as WK$_1$, WK$_2$, and so forth. The M^2-FFD will then assign each energy cost in the current order of I one by one as follows.

To assign an energy cost e_j, find

(1) the worker with the least index t such that S(WK$_t$) + $e_j \le E_t$, where S(WK$_q$) be the total sum of energy costs assigned to worker q, or S(WK$_q$) = $\sum e_j$ for all $j \in$ WK$_q$ in all work periods, and

(2) any work period k (a) that worker t is still free, and (b) to which the other copies of e_j from the same task have never been assigned.

Then, assign e_j to worker t in period k, and remove e_j from I. Continue assigning the next energy costs until I is empty. The minimum number of workers generated by M^2-FFD is set as the largest index of the workers required in the assignment.

M²-LPT Swap Heuristic

The M^2-LPT swap heuristic firstly sets the number of workers to a value, say UB. Then, it tries to gradually decrease UB one by one and search for a feasible set of work assignments for the UB workers. If the search cannot find any feasible set, the M^2-LPT swap heuristic stops. If the feasible set is found, the heuristic continues to decrease UB and search for a new feasible set until the heuristic fails or reaches an optimal solution (i.e., UB = LB). The M^2-LPT swap heuristic consists of three consecutive sub-algorithms: (1) M^2-LPT algorithm, (2) swap algorithms, and (3) multi-start algorithm. Each of these three sub-algorithms tries to identify a feasible set with UB workers. If any of them finds the feasible set, UB is decreased by one worker. Then, the solution procedure is repeated.

M²-LPT algorithm

The M^2-LPT algorithm is implemented as follows. Construct a set I as described in the M^2-FFD heuristic. From a given set of m workers, arrange them in non-increasing order of the working energy capacity and index them as WK$_1$, WK$_2$,..., WK$_{\text{UB}}$, where UB is the current number of workers. Let $R_i = E_i$ - S(WK$_i$), where R_i is the residual working energy capacity of worker i. The M^2-LPT algorithm will assign energy costs in I one by one as follows. To assign an energy cost e_j, find (1) *any* worker t with the *largest* R_t among the UB workers, and (2) any work period k (2.a) that worker t is still free and (2.b) to which the other copies of e_j from the same task have never been assigned. Then, assign e_j to worker t in period k. Remove e_j from I and continue assigning the energy costs until I is empty. Let $R_{\min} = \min\{R_1,\ldots, R_{\text{UB}}\}$. If $R_{\min}$ is less than zero, continue to the swap algorithms; otherwise, the feasible set of work assignments for the UB workers is found.

Swap algorithms

The swap algorithms is intended to swap or exchange the energy costs assigned to the same period among the UB workers so that $R_{\min}$ is greater than or equal to zero. For any p periods, there are p swap algorithms which will be applied consecutively. Half of the p swap algorithms are for increasing $R_{\min}$ while the other half are for decreasing $R_{\max}$ where $R_{\max} = \max\{R_1,\ldots, R_{\text{UB}}\}$. The swap algorithms are described below. Let $e(i, k)$ be the value of the energy cost that is currently assigned to worker i in period k.

<u>r-period swap for increasing $R_{\min}$ (r = 1 to $p/2$)</u>

1. Let any worker $i_{\min}$ be the worker whose $R_{i\min}$ is the current *minimum*.
2. Find all C_r^p possible combinations of r periods. Let **S** be a set of all combinations s_u's such that **S** = {s_u: u = 1,..., C_r^p }, where each s_u represents a combination of r periods.
3. For each s_u, consider all periods k_a's where $a \in s_u$.
4. Find any worker i_o where $i_o \ne i_{\min}$ such that the inequalities 2 and 3 hold.

$$R_{io} + [\sum_{a\in s_u} e(i_o,k_a)] - [\sum_{a\in s_u} e(i_{\min},k_a)] > R_{i\min} \tag{2}$$

$$\sum_{a\in s_u} e(i_o,k_a) < \sum_{a\in s_u} e(i_{\min},k_a) \tag{3}$$

5. If there is such worker i_o, then swap $e(i_o, k_a)$ and $e(i_{\min}, k_a)$ in workers i_o and $i_{\min}$ in all periods k_a's where $a \in s_u$.

6. Repeat steps 3 – 5 for " $s_u \in$ **S**.

<u>r-period swap for decreasing $R_{\max}$ (r = 1 to p/2)</u>

1. Let any worker $i_{\max}$ be the worker whose $R_{i\max}$ is the current *maximum*.

2. Find all C_r^p possible combinations of r periods. Let **S** be a set of all combinations s_u's such that **S** = $\{s_u$: $u = 1,\ldots, C_r^p\}$, where each s_u represents each combination of r periods.

3. For each s_u, consider all periods k_a's where $a \in s_u$.

4. Find any worker i_o where $i_o \neq i_{\max}$ such that the inequalities 4 and 5 hold.

$$R_{io} + [\sum_{a \in s_u} e(i_o, k_a)] - [\sum_{a \in s_u} e(i_{\max}, k_a)] < R_{i\max} \quad (4)$$

$$\sum_{a \in s_u} e(i_o, k_a) > \sum_{a \in s_u} e(i_{\max}, k_a) \quad (5)$$

5. If there is such worker i_o, then swap $e(i_o, k_a)$ and $e(i_{\max}, k_a)$ in workers i_o and $i_{\max}$ in all periods k_a's where $a \in s_u$.

6. Repeat steps 3 – 5 for " $s_u \in$ **S**.

After the swaps, if $R_{\min}$ is still less than zero, continue to the multi-start algorithm; otherwise, the feasible set of work assignments for the UB workers is found.

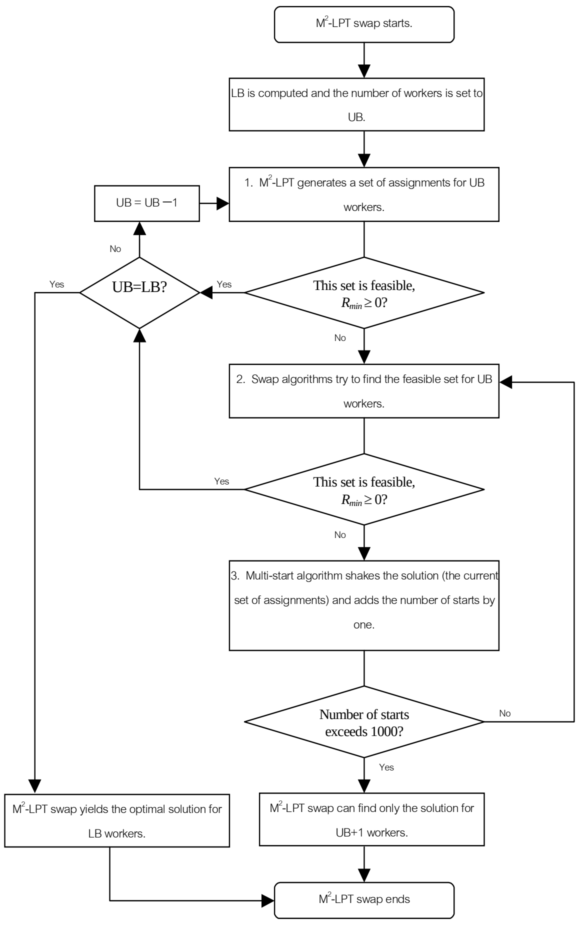
Multi-start algorithm

In this step, the current work assignment solution will be *shaken* and will re-enter the swap algorithms. To *shake* the solution, randomly select *one* pair of workers in each of all periods and swap their energy costs. Then, the resulting work assignments will be improved by the swap algorithms again. This is called a start. The number of starts is set to 1,000. While applying the multi-start algorithm, if $R_{\min} \geq 0$, the algorithm stops and checks whether UB = LB. After the algorithm has tried all 1,000 starts but $R_{\min}$ is still less than zero, the overall search fails to find a feasible set of work assignments for the UB workers and is, thus, terminated.

The flow chart in Fig. 1 summarizes the implementation of the M²-LPT swap heuristic.

Dominant Assignment (DA) Method

The dominant assignment (DA) method is a branch-and-bound method designed specially for solving WSP-E to optimality. The algorithm applies the reduction procedure along with the dominance rule proposed by Martello and Toth.[20] Let I be a set of energy costs consisting of p copies of each e_j for all tasks. An *assignment* contains at most p energy costs from I that are assigned to a worker. Let $f(i)$ be a feasible assignment for worker i if $\sum_{\forall j \in f(i)} e_j \leq E_i$.



M²-LPT swap starts.
LB is computed and the number of workers is set to UB.
1. M²-LPT generates a set of assignments for UB workers.
UB = UB −1
No
Yes
UB=LB?
Yes
This set is feasible, $R_{min} \geq 0$?
No
2. Swap algorithms try to find the feasible set for UB workers.
Yes
This set is feasible, $R_{min} \geq 0$?
No
3. Multi-start algorithm shakes the solution (the current set of assignments) and adds the number of starts by one.
Number of starts exceeds 1000?
No
Yes
M²-LPT swap yields the optimal solution for LB workers.
M²-LPT swap can find only the solution for UB+1 workers.
M²-LPT swap ends

Fig 1. Implementation of the M²-LPT swap heuristic.

$\boldsymbol{F}_i$ is the set containing all possible and distinctive $f(i)$'s; that is, $\boldsymbol{F}_i = \{f_1(i), f_2(i),\ldots, f_v(i)\}$. Given two different $f_1(i)$ and $f_2(i)$, let $Y(f_1(i))$ be the number of workers in the optimal solution obtained by forcing the solution to always include the assignment $f_1(i)$. Then, we say that $f_1(i)$ dominates $f_2(i)$ if $Y(f_1(i)) \leq Y(f_2(i))$. According to Martello and Toth[20], $f_1(i)$ dominates $f_2(i)$ if there exists a partition of $f_2(i)$ into subsets $P_{(1)},\ldots, P_{(l)}$ and a subset $\{j_{(1)},\ldots, j_{(l)}\}$ of $f_1(i)$ such that $e_{j(h)} \geq \sum_{k \in P(h)} e_k$ for $h = 1,., l$.

Let $\boldsymbol{F}_i^*$ be the subset of $\boldsymbol{F}_i$ that contains any $f_w(i) \in \boldsymbol{F}_i$ that *cannot* be dominated by the other $f(i) \in \boldsymbol{F}_i$ except $f_w(i)$ itself. The DA is implemented as follows. Let $E_1 \geq E_2 \geq \ldots \geq E_m$, for $i = 1,\ldots, m$. From the root node (or Node 0 of Level 0), calculate a lower bound (LB) from I. Construct $\boldsymbol{F}_1^*$ from all energy costs in I. Each branch of Node 0 is each assignment $f(1) \in \boldsymbol{F}_1^*$, and leads to each *son* node of Node 0 at Level 1. For any Node t of Level i (worker i), let $b = i + 1$. Construct $\boldsymbol{F}_b^*$ from all unassigned weights in I. Each branch from Node t is each assignment $f(b) \in \boldsymbol{F}_b^*$.

The bounding rule at each Node t of Level i is as follows. Let UB be the minimum number of workers known thus far. Let L(t) be the lower bound of Node t of Level i. L(t) is calculated using the following equation by considering only the unassigned energy costs of the remaining I where $d \le m$.

$$L(t) = i + \min\left\{ d \,\middle|\, \sum_{h=i+1}^{d} E_h \ge \sum_{j \in I} e_j \right\} \qquad (6)$$

Node t will be fathomed if UB ≤ L(t). At any Node t whenever all energy costs in I have already been assigned or L(t) = i and UB > L(t), then a new UB is set to L(t). Whenever UB = LB, DA terminates with the optimal solution being LB workers.

Hybrid procedure

As stated in the previous section, each of the three algorithms can be utilized separately to solve WSP-E. The work-task-period assignment solution that is obtained from either one of the two approximation algorithms (M^2-FFD and M^2-LPT swap) is guaranteed to be optimal only when UB = LB. However, the assignment solution obtained from the exact algorithm (DA) is always guaranteed to be optimal. Depending on the problem size, DA may take a few seconds or hours of computation time to yield the optimal solution. A time limit is normally set and the computation is terminated if the time limit is exceeded. If an initial UB is set too high, DA may not be able to reach an optimal solution within the given time limit. To improve the computation, a hybrid procedure is proposed.

The hybrid procedure consists of three algorithms, i.e., M^2-FFD, M^2-LPT swap and DA, which are to be applied consecutively. The procedure quickly finds a good UB and either tries to decrease UB until UB is equal to LB or proves that the solution with the current UB workers is in fact optimal and the known LB is too weak and infeasible. The proposed hybrid procedure is implemented as follows.

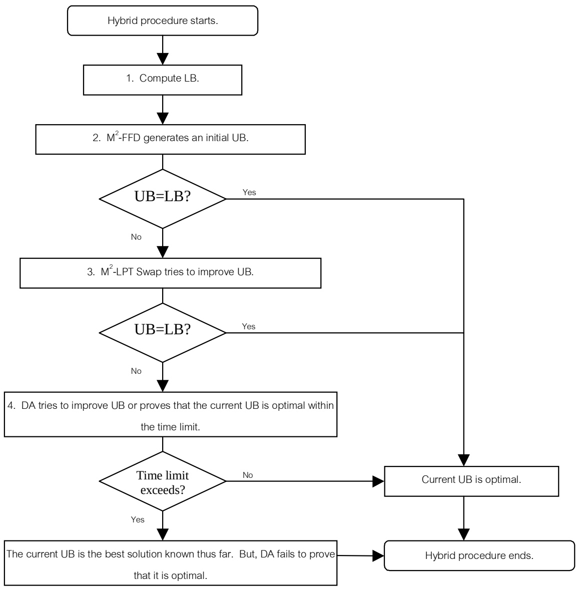
Step 1: Compute LB using Eq. (1).

Step 2: Define an initial UB using M^2-FFD.

Step 3: Apply the dual strategy using M^2-LPT swap. If the heuristic can find a feasible work assignment solution ($R_{\min} \ge 0$) for the initial UB workers, then decrease UB and find a new feasible work assignment solution. Continue to decrease UB until UB = LB or the heuristic fails to find a feasible solution.

Step 4: Improve the current UB by DA. Stop if UB = LB or the DA method can guarantee the optimality of the current UB.

Using the approximation algorithm to successively improve UB (to reduce UB), DA will not need a long computational time to verify the optimality of the work assignment solution. It is expected that the hybrid procedure will outperform each of the three algorithms if each algorithm is utilized separately. The flow chart in Fig. 2 summarizes the implementation of the hybrid procedure.



Hybrid procedure starts.
1. Compute LB.
2. M²-FFD generates an initial UB.
UB=LB?
Yes
No
3. M²-LPT Swap tries to improve UB.
UB=LB?
Yes
No
4. DA tries to improve UB or proves that the current UB is optimal within the time limit.
Time limit exceeds?
No
Yes
Current UB is optimal.
The current UB is the best solution known thus far. But, DA fails to prove that it is optimal.
Hybrid procedure ends.

Fig 2. Implementation of the hybrid procedure.

Numerical Example

Let us consider the following example. Suppose that three physical tasks (A, B, and C) are to be performed in an 8-hour day that is divided into four equal work periods. Energy costs per work period, e_j, of the three tasks are 1100, 700, and 600 kilocalories (kcal/2h), respectively. Also, suppose that there are five available workers (W1, W2, W3, W4, and W5) to be assigned to the three tasks. The working energy capacities, E_i, of the five workers are 2800, 2700, 2500, 2200, and 1800 kcal/8h, respectively.

Using Eq. (1) with m = 5, n = 3, and p = 4, we have LB = 4 workers.

Solution Based on the M^2-FFD Heuristic

A list I is constructed as I = {1100, 1100, 1100, 1100, 700, 700, 700, 700, 600, 600, 600, 600}. M^2-FFD yields the set of daily work assignments for five workers (i.e., UB = 5) as shown in Table 1. The superscript represents the order of assignment. Whenever it is infeasible to assign any energy cost to the existing workers, a new worker is added and that energy cost can be assigned to the new worker in any feasible period. Since UB > LB, it cannot be guaranteed that the minimum number of workers for this problem is five workers. The assignment solution is shown in Table 2.

Table 1. Assignment of energy costs to workers by M^2-FFD.

Worker	Work Period				Energy (kcal/8h)	
	1	2	3	4	Cost	Capacity
WK$_1$	1100^1	1100^2	600^9	-	2800	2800
WK$_2$	-	-	1100^3	1100^4	2200	2700
WK$_3$	700^5	700^6	700^7	-	2100	2500
WK$_4$	600^{10}	600^{11}	-	700^8	1900	2200
WK$_5$	-	-	-	600^{12}	600	1800

Table 2. The assignment solution obtained from M^2-FFD (five workers).

Worker	Work Period				Energy (kcal/8h)	
	1	2	3	4	Cost	Capacity
WK$_1$	A	A	C	-	2800	2800
WK$_2$	-	-	A	A	2200	2700
WK$_3$	B	B	B	-	2100	2500
WK$_4$	C	C	-	B	1900	2200
WK$_5$	-	-	-	C	600	1800

Table 3. Assignment of energy costs to four workers using the M^2-LPT algorithm.

Worker	Work Period				Energy (kcal/8h)		
	1	2	3	4	E_i	S(WK$_i$)	R_i
WK$_1$	1100^1	700^5	600^9	-	2800	2400	400
WK$_2$	700^6	1100^2	-	600^{10}	2700	2400	300
WK$_3$	600^{11}	-	1100^3	700^7	2500	2400	100
WK$_4$	-	600^{12}	700^8	1100^4	2200	2400	-200

Solution Based on the M^2-LPT Swap Heuristic

Next, the M^2-LPT algorithm finds the solution when UB = 4. Table 3 shows the assignment of energy costs to four workers as generated by the M^2-LPT algorithm.

Clearly, the resulting assignment solution is infeasible since $R_{\min}$ is still less than zero ($R_4 = -200$). It is found that the 2-period swap algorithm can increase $R_{\min}$ by swapping energy costs of workers 2 and 4 between periods 2 and 3. That is,

$R_2 + e(2, 2) + e(2, 3) - e(4, 2) - e(4, 3) \;>\; R_4$

or $\quad 300 + 1100 + 0 - 600 - 700 \;>\; -200$

After swapping $e(2, 2)$ and $e(4, 2)$ and swapping $e(2, 3)$ and $e(4, 3)$, R_4 is still $R_{\min}$ but is now equal to zero. Thus, the feasible set of assignments for four workers is found and is optimal since UB = LB = 4. Readers should note that the multi-start algorithm is unnecessary since the optimal solution was already found. Table 4 shows the optimal assignment solution for the four workers.

Table 4. The optimal work-task-period assignment solution obtained from M^2-LPT swap (four workers).

Worker	Work Period				Energy (kcal/8h)		
	1	2	3	4	E_i	S(WK$_i$)	R_i
WK$_1$	A	B	C	-	2800	2400	400
WK$_2$	B	C	B	C	2700	2600	100
WK$_3$	C	-	A	B	2500	2400	100
WK$_4$	-	A	-	A	2200	2200	0

Solution Based on the Dominant Assignment (DA) Method

To demonstrate DA, let us solve this numerical example once again from the beginning. As before, $I = \{1100^1, 1100^2, 1100^3, 1100^4, 700^1, 700^2, 700^3, 700^4, 600^1, 600^2, 600^3, 600^4\}$. The superscript indicates the order of energy cost in its set of four copies. Thus, 1100^1 is the first of the four copies of the energy cost of 1100. Note that energy costs from the same task cannot be assigned to any other worker in the same period. Figure 3 shows the decision tree generated by DA.

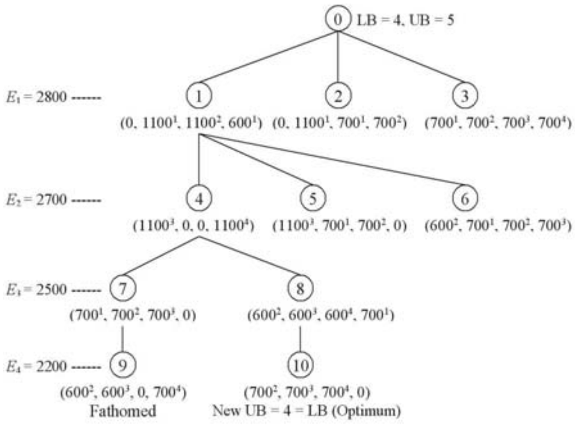


Fig 3. Decision tree generated by DA.

Level 0 ($t = 0$)

At the root node or Node 0 ($t = 0$), LB = 4. Although knowing that the optimal solution for this example is four workers, let us use the solution (five workers) from M^2-FFD as the initial UB for DA. At Node 0 in Level 1 (worker 1 whose $E_1 = 2800$) and from all energy costs in I, $\boldsymbol{F}_1^*$ can be constructed as follows.

$\boldsymbol{F}_1^* \;=\; \{(0, 1100^1, 1100^2, 600^1), (0, 1100^1, 700^1, 700^2), (700^1, 700^2, 700^3, 700^4)\}$

It is seen that assignment $(0, 1100^1, 1100^2, 600^1)$ *cannot* dominate assignments $(0, 1100^1, 700^1, 700^2)$ and $(700^1, 700^2, 700^3, 700^4)$, and vice versa. Thus, there are 3 son nodes for Node 0, which are Nodes 1, 2, and 3, respectively.

Level 1 ($i = 1, t = 1$)

At $i = 1$ and $t = 1$, remove 1100^1, 1100^2, and 600^1 from I. From Eq. (6), L(1) = 4. Since UB > L(1), Node 1 *cannot* be fathomed and the method continues. Next, construct $\boldsymbol{F}_2^*$ at Node 1 from the remaining I, where $E_2 = 2700$.

$\boldsymbol{F}_2^* = \{(1100^3, 0, 0, 1100^4), (1100^3, 700^1, 700^2, 0), (600^2, 700^1, 700^2, 700^3)\}$

Thus, there are 3 son nodes of Node 1, which are Nodes 4, 5, and 6, respectively.

Level 2 ($i = 2, t = 4$)

At $i = 2$ and $t = 4$, remove 1100^3 and 1100^4 from I. From Eq. (6), L(4) = 4. Since UB > L(4), Node 4 *cannot* be fathomed. Next, construct $\boldsymbol{F}_3^*$ at Node 4 from the remaining I, where $E_3 = 2500$.

$\boldsymbol{F}_3^* = \{(700^1, 700^2, 700^3, 0), (600^2, 600^3, 600^4, 700^1)\}$

Thus, there are 2 son nodes of Node 4, which are Nodes 7 and 8, respectively.

Level 3 ($i = 3, t = 7$)

At $i = 3$ and $t = 7$, remove 700^1, 700^2, and 700^3 from I. From Eq. (6), L(7) = 4. Once again, since UB > L(7), Node 7 *cannot* be fathomed. Next, construct $\boldsymbol{F}_4^*$ at Node 7 from the remaining I, where $E_4 = 2200$.

$\boldsymbol{F}_4^* = \{(600^2, 600^3, 0, 700^4)\}$

There is only 1 son node of Node 7, which is Node 9.

Level 4 ($i = 4, t = 9$)

At $i = 4$ and $t = 9$, remove 600^2, 600^3, and 700^4 from I. From Eq. (6), L(9) = 5. Node 9 is *fathomed* since UB = L(9). Also, Node 7 at $i = 3$ is fathomed.

Level 3 ($i = 3, t = 8$)

At $i = 3$ and $t = 8$, remove 600^2, 600^3, 600^4, and 700^1 from I. From Eq. (6), L(8) = 4. Since UB > L(8), Node 8 *cannot* be fathomed. Next, construct $\boldsymbol{F}_4^*$ at Node 8 from the remaining I, where $E_4 = 2200$.

$\boldsymbol{F}_4^* = \{(700^2, 700^3, 700^4, 0)\}$

There is only 1 son node of Node 8, which is Node 10.

Level 4 ($i = 4, t = 10$)

At Node 10 where $i = 4$ and $t = 10$, remove 700^2, 700^3, and 700^4 from I. From Eq. (6), L(10) = 4. Set I now becomes an empty set and no son node can be further generated. A new UB is then set to L(10), i.e., UB = 4. Since the new UB = 4 = LB, the optimal solution (with LB workers) is found at Node 10. Table 5 shows the optimal assignment solution generated by the DA method. From Tables 4 and 5, readers should note that the two optimal assignment solutions while yielding the same number of workers (four workers) have different work assignments.

Table 5. The optimal work-task-period assignment solution obtained from DA .

Worker	Work Period				Energy (kcal/8h)		
	1	2	3	4	E_i	S(WK$_i$)	R_i
WK$_1$	-	A	A	C	2800	2800	0
WK$_2$	A	-	-	A	2700	2200	500
WK$_3$	B	C	C	B	2500	2500	0
WK$_4$	C	B	B	-	2200	2100	100

Computational Experiment and Results

Test Problems

Three sets (A, B, and C) of test problems (WSP-E) were randomly generated. Each set consisted of 100 problems which were divided into five levels of the number of tasks (n), i.e., 10, 20, 30, 40, and 50 tasks, respectively. For each n, there were 20 test problems. The number of work periods per day p was assumed to be four equal periods. In our experiment, we considered physical tasks ranging from moderate to very heavy.[21] Energy costs e_j's were randomly generated using a uniform distribution between these following ranges:

Set A: $e_j \sim [300, 900]$ – moderate to heavy
Set B: $e_j \sim [600, 1200]$ – heavy to very heavy
Set C: $e_j \sim [300, 1200]$ – moderate to very heavy

A worker population of 1,000 male workers was then generated and the available workforce for each test problem was randomly drawn from that worker population. The daily energy capacities of the 1,000 male workers were generated using a normal distribution with its mean and standard deviation of 2400 and 243.2 kcal/8h, respectively.[5] To generate an initial number of workers m that is sufficient and feasible for each test problem, the 5th percentile energy capacity (12.5 kcal/min or 2000 kcal/8h) was used as a representative for a worker's energy capacity. The initial number of workers could be computed from the following formula.

$$m = \frac{p}{2000} \sum_{j=1}^{n} e_j \qquad (7)$$

To select workers for job rotation, m workers were randomly chosen from the 1000-worker population.

Experiment

All three algorithms and the hybrid procedure were coded in Visual Basic Application on Microsoft Excel and run on a 2.66 GHz, 512 MB RAM, Pentium IV personal computer. Each of the 300 test problems was solved by each of the three algorithms and the hybrid procedure with the same initial number of workers m. The computation time was measured in seconds. The time limit for DA when used either separately or as part of the hybrid procedure was set at 1,000 seconds. The algorithm (and the hybrid procedure) is said to have solved the test problem to optimality when (1) UB = LB, or (2) the DA method can find the optimal solution within the given time limit.

Results

Table 6 shows the number of test problems that each algorithm can guarantee an optimal solution by itself. When considering individual algorithms separately, it is seen that M^2-LPT swap is the most efficient heuristic since it could guarantee the optimality for 252 problems (or 84.00%). M^2-FFD is the least efficient heuristic, with only 9 optimal problems guaranteed. DA could guarantee only 53 problems perhaps because it has reached a 1000-second time limit before it could prove the optimality. As expected, the hybrid procedure is superior to any of the three algorithms when they are utilized separately. The hybrid procedure could find an optimal solution for 263 test problems (or 87.67%) but failed in the remaining 37 problems within the 1000-second time limit.

Table 6. The number of optimal solutions guaranteed by each solution procedure.

Set	M^2-FFD	M^2-LPT Swap	DA		Hybrid Procedure	
	(UB=LB)	(UB=LB)	(UB=LB)	(BnB*)	(UB=LB	(BnB*)
A(100)	7	99	12	0	99	0
B(100)	1	65	10	6	65	11
C(100)	1	88	25	0	88	0
Total	9	252	47 + 6 = 53		252 + 11 = 263	

*BnB means that LB is too weak and the current best UB is optimal.

It is of interest to further investigate how the three algorithms and the hybrid procedure perform in each problem set with respect to the problem size. Table 6 shows the detailed summary of the results. Specifically, it shows the number of test problems that each algorithm can reach the best solution known (called *hits*), whether or not it (the algorithm) can guarantee if the solution is an optimal solution. A breakdown of numbers of hits shows the efficiency of individual algorithms in each problem size (five levels) of each set of the test problems. It is seen that M^2-LPT swap was able to score 263 hits since there were additional 11 test problems that it could not guarantee the optimality but the lowest UB is in fact the optimal solution (as later verified by DA). Furthermore, the maximum differences (max dif) between the lowest UB generated by each algorithm and LB for individual solutions are presented in Table 7.

When DA is unable to verify the optimality, there will be a difference between UB and LB. In fact, the lowest UB could be optimal and LB is infeasible. However, the hybrid procedure fails to confirm that LB is infeasible.

The maximum differences between UB and LB from M^2-FFD, M^2-LPT swap, DA method, and the hybrid procedure are 23, 3, 29, and 3, respectively. Once

Table 7. Detailed summary of hits and maximum difference between UB and LB .

Set	*n*	Rep	Algorithm When Used Separately						Hybrid Procedure	
			M^2-FFD		M^2-LPT Swap		DA			
			Hit	Max Dif	Hit	Max Dif	Hit	Max Dif	Hit	Max Dif
	10	20	4	2	20	0	12	1	20	0
	20	20	2	4	20	0	0	7	20	0
A	30	20	1	5	19	1	0	9	19	1
	40	20	0	6	20	0	0	11	20	0
	50	20	0	6	20	0	0	17	20	0
	10	20	1	4	18	1	16	1	18	1
	20	20	0	8	18	2	0	5	18	2
B	30	20	0	15	15	3	0	15	15	3
	40	20	0	17	15	2	0	22	15	2
	50	20	0	23	10	1	0	29	10	1
C	10	20	1	2	19	1	18	1	19	1
	20	20	0	4	18	1	7	7	18	1
	30	20	0	6	17	1	0	11	17	1
	40	20	0	7	19	1	0	18	19	1
	50	20	0	7	15	1	0	23	15	1
Total Hit			9		263		53		263	
Max Dif			23		3		29		3	

again, it is seen that the hybrid procedure outperforms the other algorithms in all three problem sets. Among the 37 test problems that the hybrid procedure could not prove the optimality within the 1000-second time limit, the maximum difference between UB and LB of these 37 problems is not greater than 3. This result shows that the consecutive use of the algorithms can help to quickly improve (reduce) UB to perhaps its lowest.

From the comparison of computation times that individual algorithms and the hybrid procedure require to solve WSP-E (see Table 8), M^2-FFD is the fastest solution algorithm with its maximum computation time of only 1.0 second. M^2-LPT swap is very fast if performing only one start. With the multi-start algorithm (i.e., 1,000 re-starts), however, it required much longer time. For DA, the 1,000-second time limit was reached in all five problem sizes of the three problem sets. It is also seen that the average time of the hybrid procedure was much less than that of the DA method, owing to the consecutive use of the algorithms.

Discussion

It is seen that the hybrid procedure shows an outstanding performance in solving WSP-E. When comparing it to the other three algorithms based on the number of optimal solutions that can be guaranteed, it outperforms all of them. Of the 300 test problems, the hybrid procedure succeeded in solving 263 problems (or 87.67%), followed closely by M^2-LPT swap (252 problems or 84%), DA (53 problems or 17.67%), and M^2-FFD (9 problems or 3%). When considering the number of hits (the capability to reach the best solution known), M^2-LPT swap, although unable to verify, had also solved additional 11 problems to optimality; yielding the total of 263 hits (equal to that of the hybrid procedure). When comparing the average computation times between DA and the hybrid procedure, the use of M^2-LPT swap in the hybrid procedure to improve the initial UB yields a 78% improvement in the computation efficiency. Since M^2-LPT swap could quickly find a good UB, DA in the hybrid procedure could prove the optimality within the given time limit; thus, resulting in a much less average computation time than that of DA alone.

In each set of the 300 test problems, the hybrid procedure achieved the highest hits and could, by itself, prove that all are the optimal solutions. For the 100 problems in Set A where $e_j \sim [300, 900]$, M^2-FFD and DA performed poorly, scoring only 7 and 12 hits, respectively. M^2-LPT swap and the hybrid procedure both performed equally well irrespective of the problem size, being able to score 99 hits. This might be because most of the randomly generated e_j's are small when compared with E_i's, which would make it relatively easy to assign the tasks to workers. From the results, all 20 test problems with n = 50 in Set A have the optimal solutions (the minimum number of workers) equal to n, which also equal to LB.

Table 8. Computational times (in seconds) to find an optimal solution.

Set	n	Rep	Algorithm When Used Separately						Hybrid Procedure	
			M^2-FFD		M^2-LPT Swap		DA			
			Avg.	Max.	Avg.	Max.	Avg.	Max.	Avg.	Max.
	10	20	0.0	0.0	0.5	2.0	430.0	1000.0	1.0	2.0
	20	20	0.0	0.0	0.9	7.0	1000.0	1000.0	1.0	7.0
A	30	20	0.1	1.0	1.0	11.0	1000.0	1000.0	51.0	1011.0
	40	20	0.1	1.0	2.8	5.0	1000.0	1000.0	3.0	5.0
	50	20	0.4	1.0	4.3	7.0	1000.0	1000.0	5.0	7.0
	10	20	0.0	1.0	18.0	64.0	341.0	1000.0	187.0	1043.0
	20	20	0.0	0.0	26.9	137.0	1000.0	1000.0	129.0	1137.0
B	30	20	0.1	1.0	136.9	393.0	1000.0	1000.0	433.0	1393.0
	40	20	0.2	1.0	164.0	618.0	1000.0	1000.0	414.0	1618.0
	50	20	0.2	1.0	310.3	929.0	1000.0	1000.0	811.0	1929.0
	10	20	0.0	1.0	2.1	30.0	131.0	1000.0	52.0	1030.0
	20	20	0.0	0.0	1.3	8.0	856.0	1000.0	101.0	1008.0
C	30	20	0.1	1.0	4.6	18.0	1000.0	1000.0	155.0	1016.0
	40	20	0.1	1.0	20.6	286.0	1000.0	1000.0	190.0	1286.0
	50	20	0.2	1.0	146.4	471.0	1000.0	1000.0	397.0	1471.0
Average Time			0.1		56.0		850.7		187.3	
Maximum Time			1.0		929.0		1000.0		1929.0	

In Set B where e_j ~ [600, 1200], M^2-FFD and DA scored only 1 and 16 hits, respectively, in problems with $n = 10$. M^2-LPT swap could guarantee the optimality for 65 of the 100 test problems. With the hybrid procedure, DA could further guarantee in 11 additional test problems that the current UBs obtained by M^2-LPT swap were in fact optimal. Thus, M^2-LPT swap and the hybrid procedure could both score 76 hits. From Table 6, the problem size does have a negative effect on the performance of both solution procedures.

In Set C where e_j ~ [300, 1200], M^2-FFD could find an optimal solution of only one test problem (with n = 10) while DA could score 25 hits. Once again, M^2-LPT swap and the hybrid procedure both scored equally at 88 hits. The problem size also does have some negative effect on their performances.

The main reason why M^2-LPT swap demonstrated such an outstanding performance lies in its three algorithms: M^2-LPT, swap, and multi-start. In several cases, the swap algorithms could improve the current work assignment solution for the current UB workers such that the number of required workers could be reduced. In the multi-start algorithm where the current work assignment solution is *shaken*, only *one* pair of workers is chosen to exchange their assigned energy costs in each period. This so-called *shaking* helps to move the work assignment solution to a new point in the feasible solution space, so that the new search for an optimal solution can be re-started. It is noted that the number of re-starts is chosen to be 1,000 times based on the result from our preliminary experiment.

Since DA makes use of the branch-and-bound scheme, the computation time increases progressively with the problem size, especially when solving WSP-E. Thus, the 1000-second time limit was used in the experiment to terminate the solution search in large-sized problems. Although DA can guarantee the optimal solution when LB is too weak or infeasible, its performance decreases drastically when n is more than 10 tasks in all three problem sets.

From our observation, problems in Set A (in which the task level ranges from moderate to heavy) are very easy to solve irrespective of the problem size. This perhaps is due to the fact that most of the energy costs are comparatively smaller than one-fourth of the worker's average energy capacity (2400 kcal/8h). Problems in Set C (in which the task level ranges from moderate to very heavy) are also relatively easy to solve, perhaps for the same reason as that in Set A. For problems in Set B, the level of task ranges from heavy to very heavy. The large energy cost of the task makes it difficult to yield a work assignment for a worker that has the sum of energy costs close to his/her daily energy capacity.

M^2-LPT swap dominates M^2-FFD and DA in all three problem sets, owing to its efficient algorithms in improving (decreasing) the UB. However, it lacks the capability to guarantee the optimality of a solution when LB is too weak. The hybrid procedure combines the strengths of M^2-LPT swap and DA together and results in an efficient and also robust solution procedure.

CONCLUSION

A new workforce scheduling problem is treated in this paper. While most workforce scheduling problems aim to complete all tasks within the required time under limited resources and workers, our problem additionally considers the safety of workers. Job rotation is widely recommended when assigning workers to hazardous or physically strenuous tasks. Based on safety concerns, workers should be assigned to physical tasks with the daily sum of energy costs not more than their energy capacity to avoid over exhaustion. Our workforce scheduling problem is a variation of a well known combinatorial optimization problem, called the bin packing problem. It is however different from most bin packing problems since bin sizes in here are unequal (i.e., workers have different daily energy capacities).

Three solution algorithms (two heuristics and one exact) and a hybrid procedure are developed for solving the energy-based workforce scheduling problem (WSP-E). The problem objective is to find the minimum number of workers and their daily work assignments for a given set of physical tasks, with each worker not expending more than his/her working energy capacity. The method for finding the lower bound (LB) for WSP-E and a formula for estimating the initial upper bound (UB) are proposed. Among the two heuristics, i.e., M^2-FFD and M^2-LPT swap, the latter is much more efficient than the former due to its special algorithms to improve UB. An optimal solution is obtained when the algorithm could show that UB = LB. The dominant assignment (DA) method uses the branch-and-bound scheme to improve UB or to guarantee the optimality of UB. Lastly, a hybrid procedure that utilizes M^2-LPT swap and DA consecutively is proposed to improve the computation efficiency.

Based on the computational experiment on randomly generated 300 test problems, the hybrid procedure outperforms all three algorithms (when they are utilized separately) by obtaining an optimal solution for 263 problems. For the 37 problems in which the optimal solutions cannot be proved, the maximum difference between UB and LB is only three workers.

For future studies, the method for proving the optimality of UB can be improved in order to guarantee more problems and solve larger-sized problems efficiently. A stronger lower bound for WSP-E should be developed. Also, more practical constraints should be included in WSP-E. For example, some workers might not be able to perform certain tasks due to lack of required skills.

Acknowledgements

The authors wish to express their gratitude to the Thailand Research Fund for providing the financial support for this research study through the RGJ-PHD grant (Grant No. PHD/0138/2544). The authors also would like to thank the anonymous referees for their valuable comments.

References

1. Bhaba RS (1986) Optimum manpower models for a production system with varying production rates. *European Journal of Operational Research* **24**(3), 447-54.
2. Rangarajan N (1996) An algorithm for single shift scheduling of hierarchical workforce. *European Journal of Operational Research* **96**, 113-21.
3. Vicente V, Angeles P and Sacramento Q (1996) A graph colouring model for assigning a heterogeneous workforce to a given schedule. *European Journal of Operational Research* **90**, 285-302.
4. Vairaktaraki GL, Cai XQ and Lee CY (2002) Workforce planning in synchronous production systems. *European Journal of Operational Research* **136(3)**, 551-72.
5. National Institute for Occupational Safety and Health (NIOSH) (1981) *Work practices guide for the design of manual handling task*. Washington, DC.
6. Blazewicz J, Brauner N and Finke G (2004) Scheduling with discrete resource constraints. In: *Handbook of Scheduling: Algorithms, Models, and Performance Analysis* (Edited by Leung J Y-T), pp. 23-1 – 23-18. Chapman & Hall/CRC, Boca Raton, FL.
7. National Institute for Occupational Safety and Health (NIOSH) (1996) *National occupational research agenda*. Washington, DC.
8. National Institute for Occupational Safety and Health (NIOSH) (1997) *Musculoskeletal disorders and workplace factors (No. 97-141)*. Washington, DC.
9. Kogi K, Kawakami T, Itani T and Batino JM (2003) Low-cost work improvements that can reduce the risk of musculoskeletal disorders. *International Journal of Industrial Ergonomics* **31(3)**, 179-84.
10. Musliu N, Gärtner J and Slany W (2002) Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics* **118**, 85-98.
11. Garey MR and Johnson DS (1979) *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman and Company, San Francisco, CA.
12. Fleszar K and Hindi S (2002) New heuristics for one-dimensional bin-packing. *Computers and Operations Research* **29(7)**, 821-39.
13. Berghammer R and Reuter F (2003) A linear approximation algorithm for bin packing with absolute approximation factor 1.5. *Science of Computer Programming* **48**, 67-80.
14. Bourjolly JM and Rebetez V (2005) An analysis of lower bound procedures for the bin packing problem. *Computers and Operations Research* **32(3)**, 395-405.
15. Coffman EG-Jr, Csirik J and Woeginger GJ (1999) Approximate solutions to bin packing problems. In: *Handbook of Applied Optimization* (Edited by Pardalos PM and Resende MCG). Oxford University Press, Inc. New York, NY.
16. Friesen DK and Langston MA (1986) Variable sized bin packing. *SIAM Journal of Computing* **15**(1), 222-30.
17. Kang J and Park S (2003) Algorithms for the variable sized bin packing problem. *European Journal of Operational Research* **147(2)**, 365-72.
18. Milind D, Jayant K, and Jay S (2001) Variable sized bin packing with color constraints. *Electronic Notes in Discrete Mathematics* **7**, 1-4.
19. Johnson DS, Demers A, Ullman JD, Garey MR and Graham RL (1974) Worst-case performance bounds for simple one-dimensional packing algorithm. *SIAM Journal of Computing* **3**(4), 299-325.
20. Martello S and Toth P (1990) Lower bounds and reduction procedure for the bin packing problem. *Discrete Applied Mathematic* **28**, 59-70.
21. Astrand PO and Rodahl K (1986) *Textbook of work physiology: physiological bases of exercise*. McGraw-Hill, New York, NY.