IIE Transactions (2001) **33**, 875–887

Incorporating physical demand criteria into assembly line balancing

BRIAN J. CARNAHAN[1], BRYAN A. NORMAN[2] and MARK S. REDFERN[3]

[1]*Department of Industrial and Systems Engineering, 207 Dunstan Hall, Auburn University, AL 36849, USA*
E-mail: carnahan@eng.aubum.edu
[2]*Department of Industrial Engineering and* [3]*Department of Bioengineering, The University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA, 15261, USA*
E-mail: banorman@engrng.pitt.edu or redfern@msx.upmc.edu

Received September 1999 and accepted June 2000

Many assembly line balancing algorithms consider only task precedence and duration when minimizing cycle time. However, disregarding the physical demands of these tasks may contribute to the development of work-related musculoskeletal disorders in the assembly line workers. Three line balancing heuristics that incorporate physical demand criteria were developed to solve the problem of finding assembly line balances that consider both the time and physical demands of the assembly tasks: a ranking heuristic, a combinatorial genetic algorithm, and a problem space genetic algorithm. Each heuristic was tested using 100 assembly line balancing problems. Incorporating physical demands using these algorithms does impact the assembly line configuration. Results indicated that the problem space genetic algorithm was the most adept at finding line balances that minimized cycle time and physical workload placed upon participants. Benefits of using this approach in manufacturing environments are discussed.

1. Introduction

In facility layout design, an area of major concern for the industrial engineer is the implementation of assembly lines for manufacturing. Assembly lines can be defined as a series of manual or automated assembly workstations through which one or multiple product(s) are sequentially assembled. An example of an assembly line is shown in Fig. 1.

Figure 1 represents a balance for an I-shaped assembly line configuration. In an I-shaped line, the material flows via a conveyor or an other material handling system in a single direction. In this arrangement, a task, shown as a box with solid lines, can only be assigned to a workstation, shown as a box with broken lines, if all the predecessors of the task have been assigned to this workstation or to other workstations occurring earlier in the assembly line. Each task requires a fixed amount of processing time. The challenge of the Type-2 simple (i.e., one product) assembly line balancing problem (SALBP-2) (Salveson, 1955) is to make task assignments to a fixed number of workstations in such a way as to minimize the cycle time of the assembly operation. Cycle time, which is the inverse of the line speed, refers to the time between two consecutive products coming off the assembly line and is equal to the maximum of the workstation times. The task assignment decisions must not violate the precedence relationships that exist between the task elements.

A potential consequence of establishing a balanced assembly line operation based on task time alone is that a worker will be physically overloaded, leading to potential musculoskeletal disorders (Hagberg *et al.*, 1995). This term is used as an overall descriptor for diseases and disorders affecting the joints, muscles, tendons, ligaments, cartilage, nerves, and blood vessels of the neck and upper extremities. Work-related musculoskeletal disorders can be attributed, at least partially, to the demands of work. In regard to line balancing, the allocation of tasks to workstations may have a substantial impact on the prevalence and severity of work-related musculoskeletal disorders for the people assigned to the workstations.

The purpose of this research is to explore the incorporation of physical demand criteria in line balancing. The goal is to develop methods of assembly layout that minimize the following criteria: (i) the maximum manual gripping demands (in terms of applied force and duration) required of any worker assigned to a workstation in the assembly line; (ii) the cycle time for the assembly line operations. Three line balancing heuristics are investigated: (i) a ranking heuristic; (ii) a combinatorial genetic algorithm; (iii) and a problem space genetic algorithm. Each heuristic was designed to search for solutions that minimize both criteria simultaneously (i.e., Pareto optimization). Since the two objectives conflict with each other, it was necessary to determine relative weights for

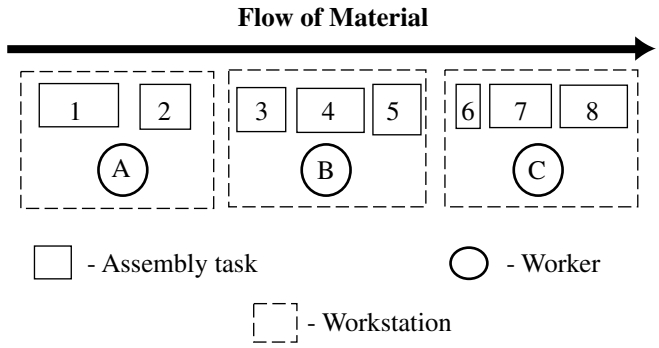

Fig. 1. Example of an I-shaped assembly line configuration.

each objective. For the purposes of this initial research, these criteria are weighted equally in each heuristic. The results of each method are contrasted.

2. The assembly line balancing problem

We now consider some of the previous research that has been conducted on assembly line balancing. First, the Type-1 assembly line balancing problem (SALBP-1) is considered. For SALBP-1, the output rate is known and the objective is to minimize the number of workstations required in order to achieve that output rate. Methods developed for SALBP-1 often serve as the basis for methods that are developed for the Type-2 problem. Following this discussion, literature pertaining to the SALBP-2 problem is considered.

2.1. *SALBP-1*

According to Scholl (1999) exact methods for solving SALBP-1 are typically variations of branch-and-bound methods and dynamic programming. Patterson and Albracht (1975) presented a 0–1 integer programming formulation of SALBP-1. The authors used a binary search procedure to identify the number of workstations that minimized the objective function. Kim and Park (1995) utilized a similar 0–1 formulation to solve an SALBP-1 in which the workstations were manned by robots. They tested their formulation on a set of problems possessing different cycle times and capacity limits. Results indicated that their approach found the optimal solution in 146 out of the 220 problems tested. Additional branch and bound methods for solving SALBP-1 include FABLE (Johnson, 1988) and EUREKA (Hoffmann, 1992). According to Scholl (1999) both methods utilize a depth first search strategy; however, EUREKA builds groups of tasks (i.e., work packages for assignment) for assignment to a single station while FABLE assigns a single task to the earliest feasible workstation in the assembly line. Scholl and Klein (1999) developed SALOME-1 which is another branch-and-bound method. SALOME-1 combines aspects of EUREKA and FABLE and adds further enhancements in order to produce an algorithm that finds better solutions than either method. In terms of dynamic programming Jackson (1956) and Held *et al.* (1963) have used this method to obtain exact solutions to relatively small problems. However, as suggested by Baybars (1986), the DP computational requirements grow exponentially with increasing problem size due to the non-polynomial complexity that is characteristic of the assembly line balancing problem. Therefore, researchers have developed heuristic approaches as a way of obtaining optimal (or near optimal) solutions. Various heuristic decision rules for solving SALBP-1 have been described by Talbot *et al.* (1986). Tabu search has also been used to solve SALBP-1 (Scholl and Voß, 1996).

2.2. *SALBP-2*

One approach to solving SALBP-2 (minimizing the cycle time for a fixed number of workstations) has been the iterative application of SALBP-1 heuristic algorithms. Hackman *et al.* (1989) outlined two main methods in which SALBP-1 approaches are used to solve SALBP-2. The first is the lower bound method which iteratively increases cycle time from its theoretical minimum until the SALBP-1 heuristic discovers a solution whose number of workstations is equal to the fixed number of stations in SALBP-2. The second approach is the upper bound method that iteratively decreases cycle time from its theoretical maximum until the SALBP-1 heuristic discovers a solution whose number of workstations is equal to the fixed number of stations in SALBP-2. A limitation to these approaches is that the analyst must know the upper and lower cycle time limits in advance.

Klein and Scholl (1996) have applied a branch-and-bound procedure to obtain solutions to SALBP-2. The authors maintain that the majority of prior research was focused on SALBP-1 and better solution procedures should be developed for SALBP-2. Identified as Simple Assembly Line Balancing Optimization Method for Type 2 (SALOME-2), this branch-and-bound procedure uses an enumeration technique in order to minimize cycle time. Coupled with the branch-and-bound procedure, SALOME-2 also uses limit rules to restrict the size of the enumeration tree to be searched. The local lower bound method of Klein and Scholl (1996) was compared to depth first search coupled with binary search, Fibonacci and binary search, and the binary search with pre-specified entry point (i.e., binary search in which the initial cycle time is set equal to the theoretical minimum). All methods were tested against 302 problem instances with the number of tasks ranging from 25 to 297. The lower bound method and the Fibonacci/binary search procedure found more optimal solutions and required less iterations than the other search techniques. However, both

of these methods found it increasingly difficult to improve on the initial feasible solution as the number of tasks increased.

Anderson and Ferris (1994) developed a Genetic Algorithm (GA) to solve an SALBP-2. The objective was to minimize the time required to complete the tasks assigned to the slowest workstation. Anderson and Ferris (1994) tested their GA against a neighborhood search scheme for problems with 50 tasks to be assigned to five workstations. The authors concluded that their GA invariably found smaller cycle times than the neighborhood search scheme. It should be noted however that the problems used by Anderson and Ferris (1994) were generated randomly, thereby preventing any meaningful comparison of their method with other heuristic approaches noted in the literature. Rubinovitz and Levitin (1995) used a GA to obtain good solutions to SALBP-2 in which the processing time for a task element was dependent upon work station assignment. The authors compared their GA to the MUltiple Solutions Technique (MUST) algorithm developed by Dar-El and Rubinovitch (1979). Both algorithms were given the problem of assigning 107 task elements to 10, 15, 20, 25, 30, and 35 workstations. For both algorithms, the average run times were recorded on each of the six problems. The GA solved the problems much faster than the MUST algorithm when the number of workstations was greater than 20. Rubinovitz and Levitin (1995) concluded that their GA approach to SALBP-2 achieved its greatest advantage when the precedence constraints were least restrictive. Fewer precedence restrictions resulted in a larger feasible region that placed exhaustive enumeration procedures, such as MUST, under computer resource limitations.

Tabu search has also been used to solve SALBP-2. In Scholl and Voß (1996) a tabu search is developed where the neighborhood consists of making swap and shift improving moves. These moves are designed to move tasks between stations in order to reduce the cycle time or if no reduction in cycle time is possible, to increase the cycle time as little as possible. A tabu list strategy is employed to prevent cycling and to drive the search away from locally optimal solutions and into unexplored regions of the search space. For further information concerning solution methods for both SALBP-1 and SALBP-2 (see Scholl, 1999).

Although various exact and heuristic approaches have been applied to the assembly line balancing problem, none of these methods have considered the physical demands placed on the workers when assigning tasks to workstations. The overriding concerns have been either: (i) minimize the number of required workstations given the cycle time; or (ii) minimize the cycle time for a given number of workstations. This disregard can lead to the development of assembly lines whose design contributes to accidents and injuries experienced by the work force. The purpose of this current research was to develop line balancing algorithms that can incorporate physical demand criteria into the task allocation process, required for solving SALBP-2.

3. Problem description

A distinctive feature of this research is that it considers worker fatigue in the assignment of tasks to assembly line workstations. To model the fatigue aspect of the problem, a physical demand component was added to each of the 100 ALB problems that were tested. The physical measure used in this study is grip strength capacity, although the proposed methodologies could also be applied to other physical measures. Grip strength capacity represents the maximum finger flexor strength generated by the worker using a semi-pronated power grip. As defined by Kroemer (1986), a power grip requires the palmer surface of the hand be in contact with the handle while enclosed by the four fingers and opposed by the thumb. To use this measure, the first step was to assign a grip strength capacity to the workers assigned to each workstation in the problem. The range of grip strength capacities ranged from 30 to 39 kg. This range is representative of the general female population and was within the limits of the female population used by Woods *et al.* (1997) to develop their grip strength fatigue model. The Woods *et al.* (1997) model will be used to evaluate the quality of proposed assembly line balances with regard to fatigue. To ensure that the full range of grip strength capacity was assigned to the workstations, the difference between the maximum and minimum grip strength (9 kg) was divided by the number of workstations (m). Starting with 30 kg, this value (V) was randomly assigned to one of the workstations and became its grip strength capacity. Once it was assigned, V was increased by $[9/(m-1)]$ and was again randomly assigned to a workstation which did not currently have a grip strength capacity. This process was repeated until all workstations had a grip strength capacity. A range of grip strength capacities are assigned to the workstations prior to assigning tasks. This is done to ensure that the line balancing algorithms used in this paper assign work packages (i.e., groups of tasks) that do not physically overload the worker. If tasks were assigned to workstations prior to worker assignment, the possibility exists that a line balancing algorithm would form work packages whose demands would exceed even the strongest of workers.

The second step was to specify the grip strength demands for each of the tasks in each of the test problems. The grip strength demands of a task vary depending on the profile of the task, the percentage of the task time spent gripping, and the force required in the gripping task. Each task possessed one of the four grip strength demand profiles illustrated in Fig. 2 where load time represents the portion of the task that requires gripping

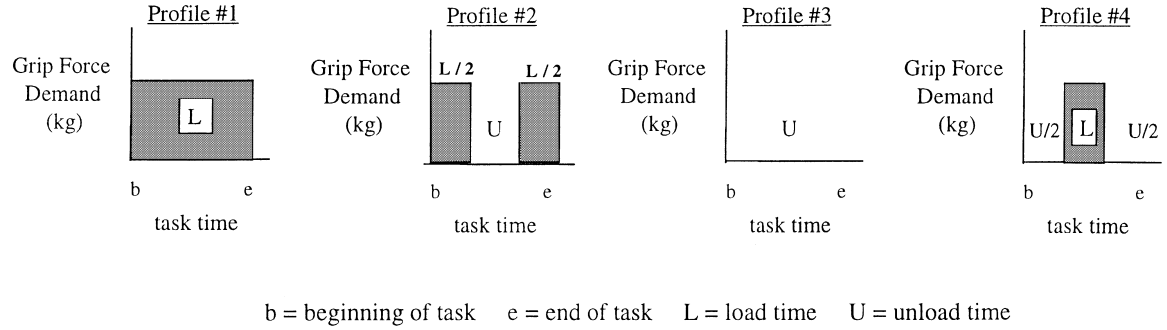

Fig. 2. Grip strength load profiles for assembly line balancing problems.

and unload time represents the task time that does not have a significant gripping component. Profile 1 represents a task that requires gripping throughout the entire task – such as using a pneumatic wrench to tighten several nuts. Profile 3 represents a task with no gripping, such as an inspection task. Profiles 2 and 4 represent tasks with both a gripping and non-gripping component.

The percentage of time spent gripping was based on industrial field studies conducted by Fransson-Hall *et al.* (1996) which found that approximately 29 to 63% of the total task time (of an assembly operation) was spent in a right-handed grip. This activity corresponds to the load time (L) in each of the profiles. An algorithm was constructed to assign one of these profiles to each task in the ALB problem. First, the percentage of each task's total processing time spent in a right-handed grip (load period, L) was determined. The load period for task i (L_i) was calculated using the following formula:

$$L_i = (\text{processing time for task } i) \times [0.29 + (0.34 \times R_i)],$$

$$R_i = \frac{S_i}{S_{\max}},$$

where S_i is the number of successors of task i and $S_{\max} = \max\{S_1, S_2, \ldots S_n\}$. Thus, for each task, the proportion of processing time spent in a right-handed grip was proportional to the task's number of successors in the precedence graph. This proportionality was established to create test problems whose most manually intensive tasks (e.g., grinding) occur "early on" in the assembly/manufacturing process. Consequently, those tasks that have few successors and occur relatively late in the assembly manufacturing process (e.g., inspection) would be less manually intensive. Note that if the load period for a task was less than 2 seconds, its load period was set equal to zero, corresponding to grip strength load profile 3. Otherwise, the task was randomly assigned grip strength profile 1, 2, or 4.

Once the load period had been established, the grip force demand for each task (G_i) was also determined. The grip force demand for each task was established using the formula:

$$G_i = 6.24 + (8.76 \times R_i).$$

The grip strength demand for a task ranged from 6.24 to 15 kg. Given the range of grip strength capacities possessed by the workers, a task required 16 to 50% of a worker's maximum grip strength capacity. This same percentage range was used to establish the grip strength fatigue equations of Woods *et al.* (1997). The approach used for assigning the L_i and G_i values assigns the largest grip strength demands to those tasks performed in the beginning of the assembly operation and assigns smaller grip strength demands to those tasks completed near the end of the assembly process. This approach was used to develop a problem set whose assembly operations resemble (in terms of manual force required) those tasks found in industry.

Three measures were selected as dependent variables for the experiments constructed in this research: (i) cycle time; (ii) fatigue (as measured by the Woods *et al.* (1997) model); and (iii) a composite measure of both cycle time and fatigue. Recall that the cycle time is equal to the maximum completion time that any one of the workstations requires to finish all its assigned tasks. To normalize this value, this variable was measured as the percent deviation from the theoretical minimum cycle time for a given problem (Scholl and Klein, 2000a).

Fatigue is defined as the decrease in an individual's capacity to do manual work. For the purposes of this research, fatigue will be measured as the percentage of grip strength retained after performing one's assigned tasks across a 4-hour shift. The grip strength fatigue model provided by Woods *et al.* (1997) was used to calculate this measure. This model is described below.

$$GE_{(i)} = GS_{(i)}(e^{-0.65(LT)})\,[(\%MVC)^{5.75} + (\%MVC - 0.22)] \quad \text{(grip strength lost)},$$

$$GS_{(i+1)} = GE_{(i)} + (MAXG - GE_{(i)})\,(1 - e^{-0.085(RT)}) \quad \text{(grip strength recovered)},$$

where: $GS_{(i)}$ = grip strength at the start of gripping interval i (kg);

$GE_{(i)}$ = grip strength at the end of gripping interval i (kg);

LT = duration of the gripping (load) interval (seconds);
$\%MVC$ = percentage of individual maximum grip strength capacity required;
$MAXG$ = individual maximum grip strength capacity (kg);
RT = duration of the rest interval (seconds).

The first equation models the loss of grip strength that occurs during a gripping interval. The second equation models the amount of grip strength recovered at the end of a rest interval. During this interval, the worker is not exerting a grip force. A sequence of gripping tasks can be evaluated by using these two equations in a cyclic fashion. Specifically, the grip strength at the end of a rest interval would also act as the grip strength at the start of the next gripping interval and so on. If one performs a series of gripping tasks, and the duration of rest time is inadequate, the grip strength of the individual at the end of a recovery period will decrease over a series of work cycles. An example of this decrease is illustrated in Fig. 3. In this figure, the grip strength at the start of the work is 38 kg. Using the two equations, the decrease in grip strength is modeled over a series of task cycles. The asymptote occurs at 31.6 kg. Using this asymptote, the percentage of grip strength retained is calculated to be $31.6/38 = 0.83$. The percentage of grip strength retained is used as the measure of fatigue and the goal is to attain a value of one. A value of one would correspond to a situation in which there was no loss of grip strength across a series of work cycles. However, reducing the amount of grip strength lost across cycles within a workstation, as predicted by the Woods *et al.* (1997) equation, requires that: (i) the required manual force (in %MVC) be reduced; (ii) the duration of static grip be reduced during the work cycle; the ratio of load time to unload time is decreased; and (iii) the placement of forceful gripping tasks back to back in a sequence be avoided. Research by Silverstein *et al.* (1987) and Viikari-Juntura *et al.* (1991) has identified forceful manual exertions as a risk factor for work related musculoskeletal disorders. In addition, Rodgers (1987) has indicated that appropriate design for repetitive work requires the ratio of unload time to load time to be at a level low enough to provide adequate recovery time to the worker (given the level of manual force required). Bird and Hill (1992) who discovered a loss in grip strength for conveyor belt workers performing repetitive work support these findings in their research. The authors identified grip weakness as a factor associated with the onset of musculoskeletal disorders within this group. In terms of the current study, the grip strength loss prediction equation is utilized as a design criterion. This criterion encourages the formation and assignment of work packages along the assembly line that limit the magnitude and duration of manual force required by workers. As seen in the literature, it is these risk factors that are associated with the development of work-related musculoskeletal disorders.

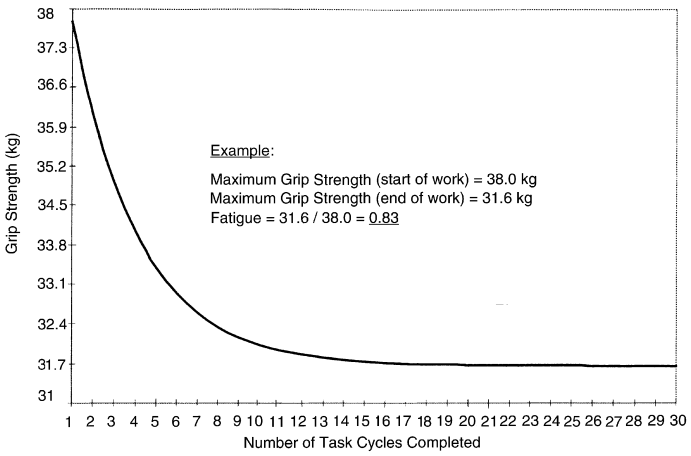

Fig. 3. The change in grip strength across task cycles.

The cycle time and fatigue measures can be combined into a single composite measure. The sum of the scaled cycle time and fatigue measures represents the composite score as shown in the following formula:

$$\text{Composite Score} = \left[\lambda\left(\frac{C}{T_{\min}}\right) + (1-\lambda)(1-\text{Min } G)\right]^{-1},$$

where: C = cycle time for the assembly operation;
$T_{\min}$ = theoretical minimum cycle time;
Min G = minimum grip strength retained across workstations;
λ = relative weighting of the cycle time objective, $1-\lambda$ is the relative weighting of the grip strength objective.

The composite score is scaled between zero and one, with the value one corresponding to the "optimal condition" (the line speed is at the theoretical maximum and the maximum percentage decrease in grip strength is zero). In this study, $\lambda = 0.5$ was used in all of the test problems.

4. Solution methodologies

We propose three solution methodologies for solving this problem. The first is a multiple ranking heuristic procedure which extends methods that have been applied to the traditional assembly line balancing problem. The second and third approaches utilize genetic algorithms to find good assembly line configurations. The two methods differ in their representation of the problem.

4.1. *Multiple ranking heuristic*

The Multiple Ranking Heuristic (MRH) is a combination of 81 separate heuristics, which utilize a composite

decision rule when determining which tasks to assign to which workstations. Each separate heuristic represents a combination of four factors: (i) method of search (lower bound, upper bound, binary); (ii) task ranking criteria (positional weight, number of successors, number of paths), (iii) composite decision rule (0.75/0.25, 0.5/0.5, 0.25/0.75); and (iv) method of assignment (forward, backward, bidirectional).

The method of search (Hackman *et al.*, 1989) manipulates the cycle time constraint that is enforced when tasks are assigned to workstations. The cycle time can be set initially to its theoretical minimum and increased one unit at a time until a feasible solution is found (lower bound search), set to its theoretical maximum and reduced one unit at a time until an infeasible solution is found (upper bound search), or the interval between the theoretical minimum and maximum can be searched using a binary search.

At any stage of the assembly line balancing procedure, more than one task may be a candidate for assignment to the current workstation. The purpose of the task ranking criteria is to prioritize this set of feasible candidate tasks with respect to either their positional weight, number of successor tasks, or number of successor paths. Based upon the criteria, a task candidate is given a task rank. The higher the task rank the more desirable the task is for assignment.

The composite decision rule is a convex combination of the task rank and the grip strength rank of each candidate task. Each task that is a candidate for assignment to a workstation has its own grip strength rank. To determine a task's grip strength rank, the multiple ranking heuristic iteratively assigns the candidate task to all feasible positions along the sequence of tasks already assigned to the workstation. For each assigned position, the percentage loss in grip strength capacity for the workstation is recorded. The algorithm records the assignment position that represents the minimal loss in grip strength. Among the task candidates, the one with the lowest minimal loss in grip strength receives the highest grip strength rank. If this task is assigned to the current workstation, it is inserted into the task sequence corresponding to its previously determined best position. As with the task rank, ties between tasks are arbitrarily broken. The candidate's composite rank is a linear combination of its task rank and its grip strength rank based on the composite decision rule selected. Ties between candidate tasks in regard to composite rank are arbitrarily broken. The task with the highest composite rank is assigned to the workstation.

The method of assignment component (Hackman *et al.*, 1989) of the MRH determines in which direction tasks are assigned to a workstation. Tasks may be assigned starting with the first workstation and ending with the last workstation (forward assignment), in a reverse order (backward assignment), or tasks can be assigned to the first and last workstation simultaneously (bidirectional assignment). The bidirectional assignment method was introduced in Scholl and Voß (1996).

This heuristic generates 81 solutions based upon unique combinations of method of search, task ranking criteria, composite decision rule, and assignment method. For every solution generated, each workstation assignment is examined to determine if slack time is present. If slack exists then the total slack time is treated as an additional task that is assigned to the workstation. Enumerative search is used to determine where this "slack" task should be placed within the workstation's task sequence so that the resulting decrease in grip strength is minimized. Once all 81 solutions have been established, the heuristic selects the solution(s) with the highest composite score.

4.2. *Combinatorial genetic algorithm*

The Combinatorial Genetic Algorithm (CGA) utilizes the Darwinian principle of natural selection to "evolve" the task allocation and task sequence of a solution to an SALBP-2. The CGA is described in terms of: (i) solution representation; (ii) fitness measure and selection scheme; and (iii) crossover and mutation mechanisms.

Each solution in the CGA is represented by an array of characters and a real number called the cycle time parameter. The array represents a First In First Out (FIFO) queue of tasks to be assigned to the workstations. The cycle time parameter represents the cycle time constraint for the assembly operation. Tasks in the queue are assigned starting with the first workstation and ending with the last workstation. If a task cannot be assigned due to the cycle time constraints or precedence restrictions, the next task in the queue is tested for feasibility of assignment. Once a task has been assigned, it is removed from the queue. This process continues until no tasks are unassigned. Once all tasks have been assigned, the cycle time is calculated and this becomes the new cycle time parameter of the solution. An example of this representation is shown in Fig. 4.

The example shown in Fig. 4 is based on the precedence graph described by Bowman (1960). The letters in the graph correspond to assembly task elements while the numbers in the graph represent the processing time for each task element. The number "28" found in the solution representation is the cycle time constraint for the assembly operation, the representation letter sequence corresponds to the order in which the tasks are assigned to the workstations. The cycle time constraint is not strictly enforced when assigning tasks to the last workstation because all of the remaining tasks are assigned to the last workstation and the total processing time of these tasks may exceed the cycle time. If the cycle time is exceeded, then the cycle time of the solution is set equal to the total time for the last workstation in order to ensure that a feasible solution is found. The total slack time for a

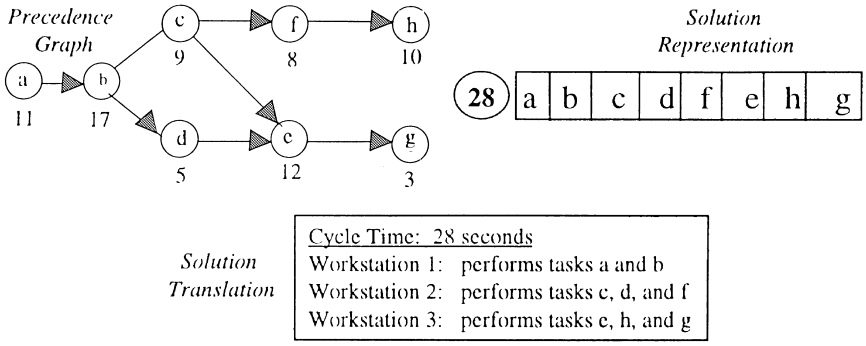

Fig. 4. Solution representation and translation for the combinatorial GA.

workstation is treated as a single slack task and the best location in the workstation task sequence is determined by enumeration as in the case of the multiple ranking heuristic.

The fitness measure for the CGA is the composite score. Under this measure of fitness, the algorithm will search for assembly line balance solutions that possess cycle times close to the theoretical minimum cycle time and have a small maximum decrease in grip strength across the workstations. When a solution found by the CGA approaches the theoretical minimum cycle time and little or no fatigue is incurred across the workstations, its fitness measure will have a value close to one. The CGA initially generates a random population of 60 solutions. In the initial population, every solution consists of a random sequence of tasks and a uniformly distributed random number chosen between the theoretical maximum and minimum cycle times. The composite score of each solution is calculated and represents its fitness. Solutions are selected for survival using stochastic selection with replacement (Goldberg, 1989). Those solutions with higher fitness values have a greater chance of contributing one or more copies of themselves to the next generation. Roulette wheel selection is used to generate 59 of the 60 solutions needed for the next generation. The most fit solution in the current population is automatically chosen for survival and a copy of it is automatically placed in the next generation.

Fragment Reordering Crossover (FRG), as described by Rubinovitz and Levitin (1995), was selected as the crossover mechanism for the task sequences of the CGA. FRG crossover allows the GA to produce offspring solutions that respect the precedence constraints of the ALB problem. An example of the FRG crossover operator is shown in Fig. 5.

Under this form of crossover, a fragment (i.e., a subsequence of tasks) of random length is selected from one of the parent solutions. The tasks within this fragment are then rearranged into the relative order in which they occur in the other solution. The offspring solutions have a combination of task allocations and sequences found in both parent solutions. In Fig. 5, the fragment consists of alleles 3, 4, and 5 from Parent 1. The order of these three alleles is changed so that they match the order found in Parent 2. Because the relative order of the alleles in the fragment comes directly from Parent 2 it must represent a feasible sequence for the fragment with respect to task precedence. Crossover is also carried out between the cycle times of the two parent solutions using arithmetical crossover (Michalewicz, 1994). The resulting offspring solutions possess attributes of both parents. Sixty percent of the selected solutions undergo crossover as seen in Suresh *et al.* (1996). The offspring solutions generated by crossover replace their parents in the surviving population. Ten percent of the selected solutions also undergo mutation. For mutation, a parent undergoes Fragment Reordering Crossover with a randomly generated sequence of tasks. In addition, the parent's cycle time parameter is also changed by means of uniform mutation (Michalewicz, 1994). The new parameter value represents the solution's new cycle time. Following mutation, the solution is reinserted back into the survivor population. The selected

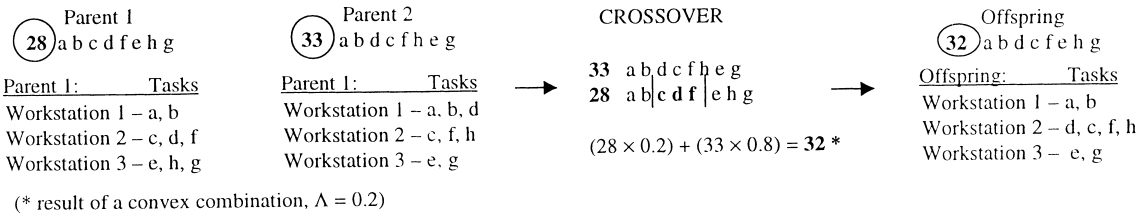

Fig. 5. An example of FRG crossover.

population in turn then replaces the original population to complete one generation cycle. This evolutionary process continues until there has not been an improvement in the most fit solution for at least 30 generations.

4.3. *The problem space genetic algorithm*

The Problem space GA (PGA) utilizes the Darwinian principle of natural selection to "evolve" a perturbation vector. As defined by Storer *et al.* (1992), a perturbation vector represents an alteration in the problem data used by a heuristic designed to solve a problem. By perturbing the original problem data, a given heuristic can find different solutions to the same problem. The PGA is described in terms of: (i) solution representation; (ii) fitness measure and selection scheme; and (iii) crossover and mutation mechanisms.

Each PGA solution is represented by an array of integers with values ranging from 0 to 2 and an array of random real numbers ranging from 1 to −1. These two arrays represent a single pass decision rule algorithm. The first array represents a strategy vector used by a ranking heuristic. The value in the first dimension of the array determines which search method is used by the algorithm (0 = lower bound, 1 = upper bound, 2 = binary). The value in the second dimension in the array specifies the assignment direction (0 = forward, 1 = backward, 2 = bidirectional). The third and final dimension of this array selects the ranking criteria used by the algorithm (0 = positional weight, 1 = number of successors, 2 = number of paths). The second array represents the perturbation vector. Each number in this array corresponds to an alteration in the ranking criteria of a single task in the assembly operation. The perturbation vector alters the ranking criteria used by the ranking heuristic. This alteration increases or decreases the rank for a particular task. An example of the solution representation is shown in Fig. 6.

As shown in Fig. 6, the single pass decision rule algorithm will use binary search coupled with a forward assignment direction. The ranks (positional weights) will be altered by way of the perturbation vector. For example, consider task b. The original positional weight is 64 but because the perturbation vector has a value of −0.25 for task b the original value is reduced by $-0.25 \times [75$ (the maximum positional weight value) − 3 (the minimum positional weight value)] to determine the perturbed value. The perturbed value is used to construct the assembly line configuration. This configuration is then evaluated using the original task data to determine its quality. If a workstation has any slack time this slack is treated as a single slack task and the best location in the workstation task sequence is determined by enumeration as in the case of the multiple ranking heuristic.

The population size and selection mechanisms are the same as those used for the CGA. Recombination of the strategy vector entails simple swapping of values between the search, assignment direction, or ranking criteria values of two solutions. Crossover between two perturbation vector solutions is carried out using an arithmetical crossover operator as described by Michalewicz (1994). An example of crossover for a six-task SALBP-2 is presented in Fig. 7.

The mating template determines the combinations of parent solutions used to yield the offspring solutions. At the start of crossover, as shown in Fig. 7, one parent is randomly chosen to be the *dominant* parent (P1). The crossover mechanism moves along the mating template. When a template value of "1" is encountered, the perturbation value of the P1 parent is altered through convex

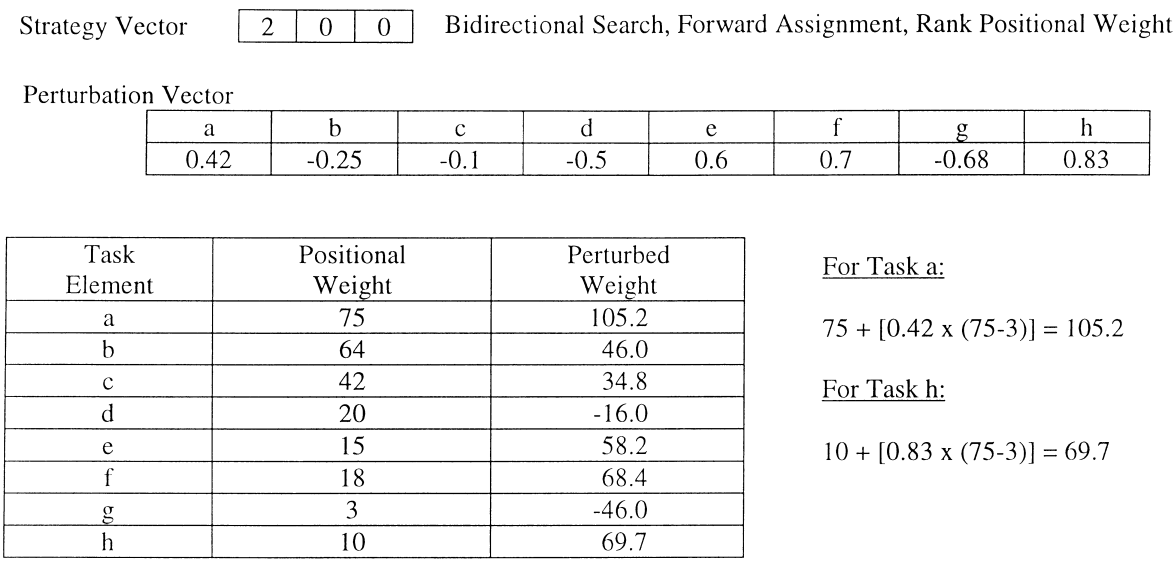

Fig. 6. Solution representation for the problem space GA.

PARENTS

P1	0.31	0.35	-0.10	0.03	0.61	0.38
	1	**0**	**0**	**1**	**1**	**0**
P2	0.42	-0.25	-0.16	0.09	0.45	0.83

Λ=0.90

OFFSPRING

0.32	0.35	-0.10	0.04	0.59	0.38
0.42	0.29	-0.11	0.09	0.45	0.43

Fig. 7. Arithmetical crossover between two perturbation vectors.

combination with the corresponding perturbation values of the P2 parent. Conversely, when a template value of "0" is encountered, the perturbation value of the P2 parent is altered through convex combination with the corresponding perturbation values of the P1 parent.

For the convex combination, the perturbation value of the dominant parent is multiplied by Λ (randomly set between 0.5 and 0.95), and the perturbation value of the non-dominant parent is multiplied by $(1 - \Lambda)$. The two offspring generated by the mating template show attributes of both parents. The number of selected solutions that undergo crossover for the problem space GA is determined in the same manner as that used in the combinatorial GA.

After selection and crossover, 10% of the survivor population undergoes mutation. The mutation operator affects both the strategy vector and the perturbation vector. For the strategy vector, the parameters affecting method of search, assignment, direction, and ranking criteria all have a 50% probability of being assigned a random value between 0 and 2. The perturbation vector is altered by arithmetic mutation (Michalewicz, 1994). Each perturbation value within the vector has a 50% probability of undergoing arithmetic mutation. Arithmetic mutation produces offspring by randomly increasing or decreasing the perturbation value. This form of mutation guarantees that all possible gene values that may have been lost in previous generations can still enter the current population to maintain solution diversity and promote search of the solution space. Once mutation is complete, the altered solution is reinserted back into the survivor population. This population in turn then replaces the original population to complete one generation cycle. This evolutionary process continues until there has not been an improvement in the most fit solution for at least 30 generations.

5. Line balancing heuristic testing and results

The three line balancing heuristics described in Section 4 (multiple ranking heuristic, combinatorial genetic algorithm, problem space genetic algorithm) were tested to determine their relative effectiveness for solving this design problem. A set of 100 Type-2 ALB problems were studied. The number of tasks in the problem set ranged from 29 to 148 (mean = 66). The number of workstations varied from 3 to 51 (mean = 17.4). The minimum cycle time (i.e., optimum) for each problem was known and ranged from 19 to 119.7 seconds (mean = 61.2 seconds). The set of 100 problems comes from Scholl (1999) and can be downloaded from Scholl and Klein (2000b). As described in Section 3, a grip strength demand component was added to the tasks in each of the 100 Type-2 ALB problems. This problem set was chosen so that it would be possible to accurately evaluate each search method's ability to solve a diverse group of line balancing problems.

To assess the impact of these line balancing heuristics on cycle time, fatigue, and composite score (i.e., solution fitness), a two factor (heuristic) × (ALB problem) Analysis Of Variance (ANOVA) model was selected. Each cell within the model contained four replications. Each replication corresponded to an individual run of a particular search method on an ALB problem. The random number seeds were used as blocks. For each replication, all three heuristics (MRH, CGA, PGA) used the same random number seed. The random number seeds were different across replications.

5.1. *Fitness evaluation across generations*

Both the CGA and PGA discover increasingly fit solutions across successive generations during a single run.

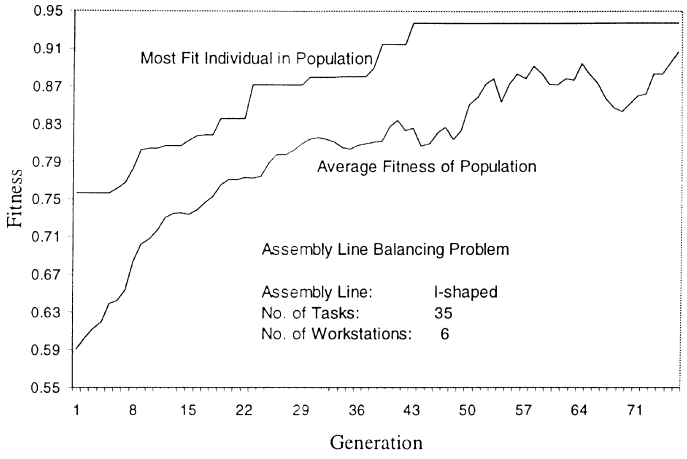

Fig. 8. Maximum and average fitness as a function of generation using the CGA.

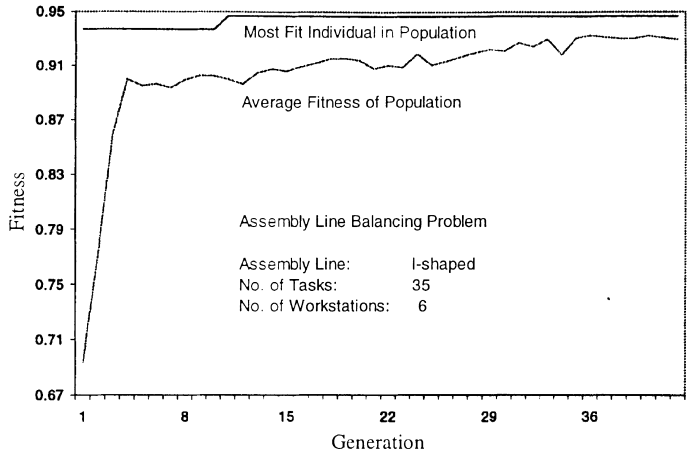

Fig. 9. Maximum and average fitness as a function of generation using the PGA.

Figures 8 and 9 plot the maximum and average fitness of the population across generations for the CGA and PGA. The average and maximum fitness values increased within the populations of solutions across generations. Both algorithms discover line balances that have shorter cycle times and are less fatiguing as the solution generations proceed.

5.2. *Differences between heuristics in terms of solution fitness*

A summary of the test problem results is provided in Table 1. Twelve different precedence graphs were studied and for each precedence graph multiple problems were developed by changing the total number of workstations that were available. Table 1 shows the average performance of each method with regard to fitness, the cycle time deviation from the lower bound (C), and the fatigue, across the different precedence graphs. The data shows that the PGA found better solutions on average than both of the other methods across all 12 of the precedence graphs.

Results from the analysis of variance revealed that in terms of fitness, the main effect of the heuristic was statistically significant with a p-value < 0.001. The Tukey post-hoc analysis test revealed that the difference in fitness between the PGA solutions and the MRH solutions was statistically significant with a p-value < 0.001. The difference in fitness between the PGA solutions and the CGA solutions was also statistically significant with a p-value < 0.001. The mean fitness values for solutions found by the CGA and the MRH were not significantly different (p-value $= 0.635$).

Results from the analysis of variance indicated that in terms of cycle time, the main effect of the heuristic was statistically significant with a p-value < 0.001. The Tukey post-hoc analysis test revealed that the differences in average cycle time between the PGA solutions and the MRH solutions and between the PGA solutions and the CGA solutions were statistically significant (p-value < 0.001). The difference in cycle times between the CGA solutions and the MRH solutions was also statistically significant (p-value $= 0.008$).

Results from the analysis of variance revealed that in terms of fatigue, the main effect of the heuristic was also statistically significant (p-value < 0.001). The Tukey post-hoc test revealed that the MRH found solutions with minimum grip strength retained values which were significantly less than the minimum grip strengths found in the solutions discovered by either the CGA or the PGA (p-value < 0.001 for both). Thus, both the CGA and PGA produced assembly line configurations with less fatigue than did the MRH. Differences in minimum grip strengths between solutions found by the CGA and the PGA were not statistically significant (p-value $= 0.974$).

Table 1. Mean values of the MRH, CGA, and PGA with respect to fitness, cycle time, and fatigue

Precedence graph	*Number of tasks*	*Number of problems*	*Multiple ranking heuristic*			*Combinatorial GA*			*Problem space GA*		
			Fitness	*C*	*Fatigue*	*Fitness*	*C*	*Fatigue*	*Fitness*	*C*	*Fatigue*
Buxey	29	8	0.811	0.089	0.846	0.840	0.086	0.892	0.855	0.072	0.899
Sawyer	30	8	0.828	0.088	0.876	0.850	0.077	0.897	0.856	0.064	0.892
Gunther	35	10	0.858	0.067	0.900	0.860	0.070	0.905	0.880	0.056	0.918
Kilbrid	45	7	0.956	0.022	0.975	0.958	0.020	0.976	0.964	0.013	0.975
Hahn	53	7	0.925	0.053	0.969	0.922	0.064	0.976	0.927	0.057	0.975
Warnecke	58	14	0.846	0.097	0.912	0.846	0.126	0.941	0.870	0.078	0.926
Wee-Mag	75	12	0.901	0.098	0.984	0.904	0.095	0.986	0.918	0.077	0.985
Arcus	83	5	0.865	0.048	0.890	0.862	0.053	0.891	0.872	0.037	0.889
Lutz 2	89	8	0.810	0.082	0.845	0.820	0.081	0.858	0.833	0.063	0.860
Lutz 3	89	9	0.846	0.065	0.881	0.834	0.089	0.887	0.860	0.066	0.902
Mukherjee	94	6	0.889	0.054	0.925	0.887	0.049	0.918	0.893	0.045	0.921
Bartholdi 2	148	6	0.892	0.052	0.930	0.879	0.117	0.979	0.912	0.050	0.953

Table 2. Tukey test results for the MRH, CGA, and PGA with respect to fitness, cycle time, and fatigue

Fitness			*Cycle time*			*Fatigue*		
MRH	*CGA*	*PGA*	*MRH*	*CGA*	*PGA*	*MRH*	*CGA*	*PGA*
———		—	—	—	—	—	———	

Table 2 summarizes these results where a solid line under two treatment levels indicates that the difference in the means between the two treatment levels is not significant at the 0.01 level.

Additional evidence advocating the use of genetic algorithms in assembly line balancing can be found when comparing the most fit ALB solutions found by the PGA and the MRH. The PGA-based solutions possessed higher fitness, shorter cycle times, and retained a higher minimum grip strength than those ALB solutions discovered by the MRH. Table 3 illustrates these differences.

Within Table 3, three SALBPs of increasing size, in terms of tasks and workstations, are presented. For all three problems, the most fit solution discovered by the PGA exceeded the most fit solution discovered by the MRH. The examples documented in Table 3 illustrate how a GA can be hybridized with more traditional search methods. The hybrid genetic algorithm (PGA) discovered ALB solutions that possessed fitness values that were significantly greater than those solutions discovered by the pure GA search (CGA) or the more traditional search (MRH). The PGA-based solutions had shorter cycle times than the MRH-based solutions and the CGA-based solutions. The PGA-based solutions were also less fatiguing than the MRH-based solutions.

Table 3. Three examples comparing the ALB solutions found by the MRH and PGA

Solution example	*Tasks*	*Work stations*	*Fitness*		*Cycle time*[1]		*Fatigue*[2]	
			MRH	*PGA*	*MRH*	*PGA*	*MRH*	*PGA*
1	29	8	0.869	0.922	6.17	1.23	91	93
2	83	22	0.856	0.898	0.84	0.51	84	89
3	148	45	0.894	0.920	6.01	3.93	94	95

[1] Refers to the percent deviation from the theoretical minimum cycle time.
[2] Refers to the minimum retained grip strength (in %*MVC*) found across workstations.

5.3. *Differences between heuristics in terms of solution fitness*

The solutions generated by using the dual objectives of cycle time and grip strength loss are different than those developed by solely considering cycle time. In some of the problems it was possible to find solutions with the same cycle time that had significantly different performance concerning grip strength loss. For example, Table 4 shows results for five problems of different sizes. Each problem was solved once using an objective function of cycle time alone and then using an objective function that combined cycle time and the grip strength criterion. For these problems, both methods found solutions with the optimal cycle time, however the grip strength loss differs significantly. In many of the other problems, the dual criteria methods sacrificed cycle time slightly in order to improve the grip strength loss performance. These results indicate the importance of considering both objectives in determining an appropriate line balance.

6. Conclusions and directions for future research

Future experiments may be designed which can further extend the current research. Given the multi-faceted nature of the problem, these research extensions could be in terms of ergonomics, safety, operations research, or manufacturing. In the current research, the weighting factors for both fatigue and cycle time were equivalent. Future projects could observe and record the differences in the solutions developed when one factor is weighted over another. From an ergonomics perspective, future

Table 4. Comparison of solutions based on using cycle time alone versus cycle time combined with grip strength loss

SALBP problem	*Number of tasks*	*Number of workstations*	*Cycle time*	*Solution set A minimum grip strength*	*Solution set B minimum grip strength*
Buxey	29	8	41	0.786 (21.4%)	0.928 (7.20%)
Gunther	35	6	84	0.894 (10.6%)	0.987 (1.30%)
Kilbrid	45	11	55	0.909 (9.10%)	0.983 (1.70%)
Wee-Mag	75	30	56	0.882 (11.8%)	0.957 (4.30%)
Lutz 2	89	11	45	0.843 (15.7%)	0.923 (7.70%)

Solution set A – Ergonomics criteria *not* used in genetic search of SALBP-2 solution.
Solution set B – Ergonomics criteria used in genetic search of SALBP-2 solution.
Minimum Grip Strength – Minimum grip strength retained across workstations (percentage of grip strength lost).

work may focus on incorporating other ergonomics criteria (e.g., awkward posture, vibration) into assembly line balancing problems. Extensions of this research could also improve industrial safety in the workplace. Future line balancing heuristics could be encoded with general or specific safety guidelines. These guidelines would prevent the heuristics from assigning potentially hazardous combinations of tasks (e.g., painting and welding) to the same worker.

Future controlled experimental designs might focus on optimizing the operating parameters (i.e., population size, the number of generations, the method of selection, types or rates of crossover, and types or rates of mutation) of the PGA and CGA. Other extensions of the current study into operations research might also focus on developing additional hybrid search algorithms to solve SALBP-2. For example, a heuristic (such as the MRH) could be used to generate the initial population of the GA (Len *et al.*, 1993). The heuristic would seed the population with initial solutions, then the GA would use evolutionary computation to continue the search. Expanding the current research to other assembly line balancing problems would also seem prudent. For example, the objective of SALBP-1 is to minimize the number of workstations for a fixed cycle time. Similar concerns, regarding worker safety, could also be the focus in SALBP-E. In this case, minimizing the product of cycle time and number of workstations could increase the number of tasks a worker must perform.

In summary, the results of the current research strongly suggest that it is possible to create assembly lines that incorporate both production concerns and physical demand criteria. Furthermore, heuristics can be constructed which systematically discover these potential ALB design solutions. The production planner can review the collection of solutions and identify those ALB designs that are best suited for a particular manufacturing operation in terms of productivity and safety.

References

Anderson, E.J. and Ferris, M.C. (1994) Genetic algorithms for combinatorial optimization: the assembly line balancing problem. *ORSA Journal on Computing*, **6**(2), 161–173.

Baybars, I. (1986) A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, **32**(8), 909–932.

Bird, H.A. and Hill, J. (1992) Repetitive strain disorder: towards diagnostic criteria. *Annals of Rheumatoid Diseases*, **51**(8), 974–977.

Bowman, E.H. (1960) Assembly-line balancing by linear programming. *Operations Research*, **8**, 385–389.

Dar-El, E.M. and Rubinovitch, Y. (1979) MUST – a multiple solutions technique for balancing single model assembly lines. *Management Science*, **25**(11), 1105–1114.

Fransson-Hall, C., Bystrom, S. and Kilbom, A. (1996) Characteristics of forearm-hand exposure in relation to symptoms among automobile assembly line workers. *American Journal of Industrial Medicine*, **29**, 15–22.

Goldberg, D.E. (1989) Genetic *Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, New York.

Hackman, S.T., Magazine, M.J. and Wee, T.S. (1989) Fast, effective algorithms for simple assembly line balancing problems. *Operations Research*, **37**(6), 916–924.

Hagberg, M., Silverstein, B., Wells, R., Smith, M. J., Hendrick, H. W., Carayon, P. and Perusse, M. (1995) *Work Related Musculoskeletal Disorders (WRMSDs): A Reference Book for Prevention*, Taylor and Francis, New York.

Held, M., Karp, R.M. and Shareshian, R. (1963) Assembly-line balancing – dynamic programming with precedence constraints. *Operations Research*, **11**, 442–459.

Hoffman, T.R. (1992) EUREKA: a hybrid system for assembly line balancing. *Management Science*, **38**, 39–47.

Jackson, J.R. (1956) A computing procedure for a line balancing problem. *Management Science*, **2**, 261–271.

Johnson, R.V. (1988) Optimally balancing large assembly lines with FABLE. *Management Science*, **34**(2), 240–253.

Klein, R. and Scholl, A. (1996) Maximizing the production rate in a simple assembly line balancing – a branch and bound procedure. *European Journal of Operations Research*, **91**, 367–385.

Kim, H. and Park, S. (1995) A strong cutting plane algorithm for the robotic assembly line balancing problem. *International Journal of Production Research*, **33**(8), 2311–2323.

Kroemer, K. (1986) Coupling the hand with the handle: an improved notation of touch, grip, and grasp. *Human Factors*, **28**(3), 337–339.

Leu, Y., Matheson, L.A. and Rees, L.P. (1993) Assembly line balancing using genetic algorithms with heuristic-generated initial populations and multiple evaluation criteria. *Decision Sciences*, **25**(4), 581–605.

Michalewicz, Z. (1994) *Genetic Algorithms + Data Structures = Evolutionary Programs*, 2nd Edn., Springer Verlag, New York.

Patterson, J.H. and Albracht, J.J. (1975) Assembly-line balancing: zero-one programming with Fibonacci search. *Operations Research*, **23**, 166–172.

Rodgers, S.H. (1987) Recovery time needs for repetitive work. *Seminars in Occupational Medicine*, **2**(1), 19–24.

Rubinovitz, J. and Levitin, G. (1995) Genetic algorithm for assembly line balancing. *International Journal of Production Economics*, **41**, 343–354.

Salveson, M.E. (1955) The assembly line balancing problem. *Journal of Industrial Engineering*, **6**(3), 18–25.

Scholl, A. (1999) *Balancing and Sequencing of Assembly Lines*, 2nd Edn., Physica-Verlag, Heidelberg, Germany.

Scholl, A. and Klein, R. (1999) Balancing assembly lines effectively – a computational comparison. *European Journal of Operational Research*, **114**, 50–58.

Scholl, A. and Klein, R. (2000a) Improving the efficiency of JIT assembly lines: the U-line balancing problem. *International Journal of Production Research*, (in press).

Scholl, A. and Klein, R. (2000b) The task times and precedence graph information for the test problems can be downloaded from: *http://www.bwl.tu-darmstadt.de/bwl3/forsch/projekte/alb/*.

Scholl, A. and Voß, S. (1996) Simple assembly line balancing – heuristic approaches. *Journal of Heuristics*, **2**(3), 217–244.

Silverstein, B.A., Fine, L.J. and Armstrong, T.J. (1987) Occupational factors and the carpal tunnel syndrome. *American Journal of Industrial Medicine*, **11**, 343–358.

Storer, R.H., Wu, D. and Vaccari, R. (1992) New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, **38**(10), 1495–1509.

Suresh, G., Vinod, V.V. and Sahu, S. (1996) A genetic algorithm for assembly line balancing. *Production Planning and Control*, **7**(1), 38–46.

Talbot, B.F., Patterson, J.H. and Gehrlein, W.V. (1986) A comparative evaluation of heuristic line balancing techniques. *Management Science*, **32**(4), 430–454.

Viikari-Juntura, E., Kurppa, K., Kuosma, E., Huuskonen, M., Kuorinka, I., Ketola, R. and Konni, U. (1991) Prevalence of epicondylitis and elbow pain in the meat processing industry. *Scandinavian Journal of Work Environment and Health*, **17**, 38–45.

Woods, D., Fisher, D.L. and Andres, R.O. (1997) Minimizing fatigue during repetitive jobs: optimal work rest schedules. *Human Factors*, **39**(1), 83–101.

Biographies

Brian Carnahan received his doctorate in Industrial Engineering at the University of Pittsburgh. This degree is added to his B.S. in Science from the Pennsylvania State University and Masters in Industrial Engineering and Operations Research from the University of Massachusetts. He has worked extensively as an industrial engineer and ergonomist for the US Department of Labor's Occupational Safety and Health Administration (OSHA) in Washington DC. In addition, he has held a position as a human factors specialist for the Carnegie Mellon Research Group in Pittsburgh and CMU's Driver Training and Safety Institute. Currently, Dr. Carnahan is an Assistant Professor in the Industrial and Systems Engineering Department at Auburn University. Dr. Carnahan's research interests have focused primarily upon developing artificial intelligence applications to help solve challenging problems in ergonomics and safety. These applications have focused on the automated design of safe lifting tasks, industrial job rotation scheduling, assembly line balancing, low back injury risk modeling, and skill and accident analyses of long-haulage truck drivers. Dr. Carnahan is a member of the Human Factors and Ergonomics Society, The Industrial Ergonomics Technical Group, The Institute for Industrial Engineers, and the Alpha Pi Mu Society.

Bryan A. Norman is an Assistant Professor of Industrial Engineering at the University of Pittsburgh. He received the Ph.D. degree in Industrial and Operations Engineering from the University of Michigan in 1995, where he was a National Science Foundation Fellowship holder, and has B.S.I.E. and M.S.I.E. degrees from the University of Oklahoma. His research interests primarily focus on the modeling of complex problems in manufacturing systems. His areas of application include scheduling, sequencing, job rotation, assembly line balancing and facility layout and material handling system design. His research has been funded by the National Science Foundation, the Ben Franklin Technology Center of Western Pennsylvania, the Society of Manufacturing Engineers and by local industry. He has published his research in *IIE Transactions*, *Naval Research Logistics*, *Engineering Design and Automation*, and *Computers and Industrial Engineering*. He is a member of IIE and INFORMS.

Mark S. Redfern obtained his Ph.D. in Bioengineering from the University of Michigan in 1988. Currently, Mark is an Associate Professor in the Department of Bioengineering, University of Pittsburgh with appointments in the Department of Industrial Engineering and the School of Medicine. He is Director of the Human Movement and Balance Laboratory where he conducts research in human postural control, ergonomics and occupational injury prevention. He has 40 peer review publications, 10 book chapters, and over 55 proceedings and meeting abstracts.

Contributed by the Applied Optimization Department