The Double-Bay Layout Problem

Xingquan Zuo, *Senior Member, IEEE*, Chase C. Murray, *Member, IEEE*, and Alice E. Smith, *Senior Member, IEEE*

***Abstract*—The layout of a semiconductor manufacturing facility can play a significant role in reducing operating costs and improving productivity. This paper introduces the double-bay layout problem that is encountered in semiconductor fabrication. This new problem not only considers the assignment and relative ordering of machines in two bays but also determines the exact location of each machine, and incorporates two objectives of minimizing material handling cost and layout area. A hybrid methodology combining a multi-objective genetic algorithm with linear programming is proposed to solve this problem. The genetic algorithm effectively identifies the set of non-dominated machine sequences, while the linear program uses the relative assignments obtained by the genetic algorithm to determine optimal absolute machine locations. Experimental results indicate that the proposed hybrid methodology finds Pareto-optimal solutions for small-scale problems and high-quality solutions for problems of practical size.**

***Index Terms*—Layout, manufacturing planning, semiconductor device manufacture.**

I. Introduction

WELL-DESIGNED facility layouts can enhance system performance by reducing operating costs and improving productivity, equipment utilization, and product yield [1]. One common facility layout class in semiconductor manufacturing is the bay configuration since it affords ready machine access for maintenance and operation of the physical equipment [2]. Bay configurations are popular in a wide-variety of manufacturing contexts, including steel production, artillery combat vehicle fabrication, and bridge crane manufacturing [3], although research on bay layout problems is scarce [4]. Moreover, existing studies consider the material handling cost as the only optimization objective, ignoring the layout area. For semiconductor manufacturing in particular, the layout area should also be considered, as cleanroom construction costs may exceed $3,500 per square foot [5]. In addition, previous studies only determine the relative ordering of machines in each bay and do not optimize the exact location of each machine. However, optimizing exact locations of machines can decrease the material handling cost for layout problems with more than one row. For industries with tight profit margins, even a small decrease in material handling cost may translate into considerable additional profit.

Semiconductor manufacturers have traditionally adopted functional area layouts, whereby all machines of the same type share the same bay. While such layouts may support high product variety or low product volumes (see [6], [7]), [8] notes that this results in high inter-bay material movement. To reduce the resulting extended lead times, improve resource utilization, and increase throughput rates, [8] proposes a distributed bay layout configuration, where replicas of a given machine type may be assigned to different bays. As such, this paper considers the distributed bay configuration, thus decreasing inter-bay material movement by allowing two bays to contain all machine types required to produce certain products.

We define the *double-bay layout problem* (DBLP) in which the two bays are linked by an overhead unidirectional track for material travel, as illustrated in Figure 1. This automated material handling system (AMHS) has been widely implemented in semiconductor fabrication due to its relatively low cost in dispatching and traffic control, and the ability to meet high-speed delivery requirements [6], [9]. Solutions to the DBLP locate each of m machines in one of the four rows defining the two bays, where $R = \{1, 2, 3, 4\}$ denotes the set of rows. Each machine $i \in I$ has a width w_i and a depth d_i, where $I = \{1, \ldots, m\}$ represents the set of all machines. It is assumed that the load/unload port of each machine is located at the midpoint of that machine's width. The number of products that must travel from machine $i \in I$ to $j \in \{I \backslash i\}$ is given by $f_{i,j}$. Each pair of machines i and j must be separated by at least a minimum clearance, denoted as a_{ij}. Note in Figure 1 machines 13 and 15 are separated by a distance greater than $a_{13,15}$. Separations exceeding the minimum clearance requirements can reduce material handling costs, as will be made clearer shortly.

Each bay (that is, the area between rows 1 and 2, or between rows 3 and 4) has a pre-specified width, c, which also defines the width of the AMHS track within each bay. The length of the AMHS track extending outside the widest bay is a constant, e. A sufficient pre-specified clearance, C, must exist between the deepest machines in rows 2 and 3 to allow equipment to be moved in or out.

In this paper, a mixed integer linear programming model for the complete DBLP is established, such that a mathematical programming solver can find its globally optimal solution for small-scale problems. To achieve Pareto optimal solutions for

Manuscript received May 3, 2015; revised February 3, 2016; accepted August 18, 2016. Date of publication August 26, 2016; date of current version October 27, 2016. This work was supported by the National Natural Science Foundation of China under Grant 61374204.

X. Zuo is with the School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: zuoxq@bupt.edu.cn; zuoxq@ieee.org).

C. C. Murray is with the Department of Industrial and Systems Engineering, University at Buffalo, Buffalo, NY 14260 USA (e-mail: cmurray3@buffalo.edu).

A. E. Smith is with the Department of Industrial and Systems Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: smithae@auburn.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSM.2016.2603443

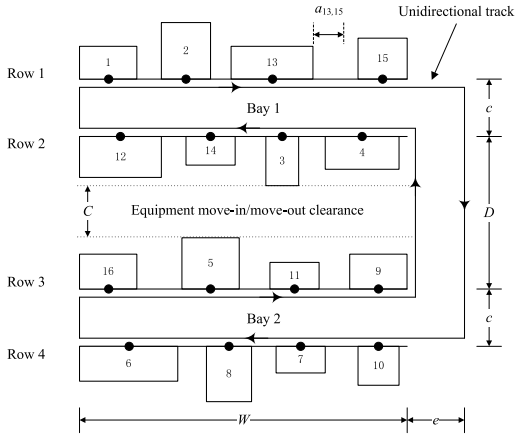

Fig. 1. A notional representation of the double-bay layout problem.

TABLE I
DECISION VARIABLES

x_{ir}	Continuous decision variable representing the location of machine $i \in I$ in row $r \in R$. If i is not placed in row r, $x_{ir} = 0$.
y_{ir}	If machine $i \in I$ is placed in row $r \in R$, $y_{ir} = 1$; otherwise, $y_{ir} = 0$.
z_{rij}	If machine $i \in I$ is placed to the left of machine $j \in \{I \setminus i\}$ in row $r \in R$, $z_{rij} = 1$; otherwise, $z_{rij} = 0$.
W	The maximum distance between the left side of the first machine in any row and the right side of the last machine in any row, not including e.
d'_2 (d'_3)	The depth of the deepest machine in row 2 (3).
D	Depth of the bay pitch, where $D = d'_2 + C + d'_3$.
L	The total length of the unidirectional track.
s_r	Area consumed by the machines in row $r \in R$.
A	Total area consumed by the layout, as determined by the area of the smallest rectangle enclosing all machines and the track.
T_{ij}	Travel distance from machine $i \in I$ to $j \in \{I \setminus i\}$ along the unidirectional track.

large-scale problems, we propose a novel hybrid methodology combining a multi-objective genetic algorithm and linear programing, which we term MOGA-LP. The MOGA simultaneously optimizes the assignment and order of machines in each bay to obtain a set of non-dominated machine sequences. Based on the formulation of the DBLP, a linear program (LP) is solved for each non-dominated machine sequence to find the optimal locations of machines in the sequence. The solutions produced by the LP form the set of final Pareto solutions.

II. Related Work

The double row layout problem (DRLP), bay layout problems, and unidirectional loop layout problem (ULLP) are related to our work and are reviewed below.

The DRLP, first introduced by [10], seeks to arrange machines on either side of one aisle, such that adjacent machines must be separated by a minimum pre-specified clearance. A mixed-integer programming formulation and five heuristics were proposed by [11], followed by a revised formulation in [12]. A local search procedure for the DRLP with asymmetric material flow was proposed by [13]. An extended DRLP (EDRLP) in which a non-zero aisle width is incorporated, and the objectives of minimizing both material handling cost and layout area were considered in a linearly-weighted objective function, was presented in [14]. A multi-objective tabu search heuristic was proposed in [15]. Similar to the DRLP, the corridor allocation problem (CAP) considers two parallel rows of machines separated by a single aisle (see [16]). However, the CAP does not consider clearance requirements between machines, making it a purely combinatorial problem.

The DBLP can be viewed as a combination of two DRLPs linked by an AMHS. However, while the DRLP assumes that materials may move directly between machines (e.g., by hand), the DBLP uses the unidirectional track for material transfer.

Bay layout problems seek the assignment of departments to multiple parallel bays in an effort to minimize the material handling costs between bays. A modified quadratic set covering problem formulation was established by [6]. An extension, in which the order of facilities in each bay is also determined, was provided by [3]. A two-stage solution methodology was proposed, whereby a mixed integer program assigned departments to bays, followed by a dynamic programming formulation to determine the ordering of departments within each bay. A similar two-stage procedure was proposed by [8] in the context of a distributed bay layout problem, where replicas of a given department type may exist in different bays. While the above studies assumed that each bay has fixed and identical widths, [4] considered bay layout problems in which limited flexibility in the areas of each bay is allowed. Recently, [17] proposed a hybrid of particle swarm optimization and local search to solve a layout problem with a flexible bay structure, where the width of each bay depends on the total area of the departments assigned therein.

By contrast, the proposed DBLP explicitly considers the material handling costs both between and within bays, where minimum clearance requirements must be observed. Furthermore, we are not aware of any multi-objective bay layout problem with the objective of minimizing layout area.

The ULLP, a review of which is provided by [18], seeks to assign *m* workstations to *m* candidate locations in a closed one-directional loop (racetrack) to minimize the total handling cost of the manufactured parts. Traditionally, one of the *m* workstations is designated to be a load/unload station in which parts enter/exit the system. However, [19] generalized this problem by considering attachable equipment to allow each workstation to serve as a load/unload point.

Although our problem utilizes a unidirectional material handling system, the DBLP differs from the ULLP in several respects. First, the ULLP does not explicitly consider machine clearance restrictions, as the *m* candidate workstation locations are pre-specified. Additionally, the workstations in a ULLP are arranged around the exterior of a racetrack-shaped conveyor system, rather than in a bay-based layout. Finally, the ULLP does not consider the minimization of total space requirements.

III. Problem Formulation and Structural Property Analysis

A mixed integer linear programming formulation of the DBLP is established, using the parameter notation defined in Section I. Decision variables are summarized in Table I.

$$\text{Min} \quad \{Obj_1, Obj_2\} \tag{1}$$

$$Obj_1 = \sum_{i\in I}\sum_{j\in\{I\setminus i\}} f_{ij}T_{ij} \tag{2}$$

$$Obj_2 = A \tag{3}$$

$$\text{s.t.} \quad W \geq x_{ir} + (1/2)w_i y_{ir} \quad \forall\ i \in I, r \in R, \tag{4}$$

$$x_{ir} - (1/2)w_i y_{ir} \geq 0 \quad \forall\ i \in I, r \in R, \tag{5}$$

$$s_r \geq d_i(W + e) - d_i M(1 - y_{ir}) \quad \forall\ r \in R, i \in I, \tag{6}$$

$$A = \sum_{r\in R} s_r + W(C + 2c) + e(C + 2c), \tag{7}$$

$$x_{ir} \leq M y_{ir} \quad \forall\ i \in I, r \in R, \tag{8}$$

$$\sum_{r\in R} y_{ir} = 1 \quad \forall\ i \in I, \tag{9}$$

$$\big(w_i y_{ir} + w_j y_{jr}\big)/2 + a_{ij}z_{rji} \leq x_{ir} - x_{jr} + M\big(1 - z_{rji}\big) \quad \forall\ i \in I^1, j \in I^2, r \in R, \tag{10}$$

$$\big(w_i y_{ir} + w_j y_{jr}\big)/2 + a_{ij}z_{rij} \leq -x_{ir} + x_{jr} + M\big(1 - z_{rij}\big) \quad \forall\ i \in I^1, j \in I^2, r \in R, \tag{11}$$

$$T_{ij} \geq x_{jr} - x_{ir} - M\big(1 - z_{rij}\big) \quad \forall\ r \in \{1, 3\}, i \in I, j \in \{I \setminus i\}, \tag{12}$$

$$T_{ij} \leq x_{jr} - x_{ir} + M\big(1 - z_{rij}\big) \quad \forall\ r \in \{1, 3\}, i \in I, j \in \{I \setminus i\}, \tag{13}$$

$$T_{ij} \geq L - x_{jr} + x_{ir} - M\big(1 - z_{rij}\big) \quad \forall\ r \in \{2, 4\}, i \in I, j \in \{I \setminus i\}, \tag{14}$$

$$T_{ij} \leq L - x_{jr} + x_{ir} + M\big(1 - z_{rij}\big) \quad \forall\ r \in \{2, 4\}, i \in I, j \in \{I \setminus i\}, \tag{15}$$

$$T_{ij} \geq L - c - x_{i1} - x_{j2} - M\big(2 - y_{i1} - y_{j2}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{16}$$

$$T_{ij} \leq L - c - x_{i1} - x_{j2} + M\big(2 - y_{i1} - y_{j2}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{17}$$

$$T_{ij} \geq 2W + 3c + 2e + D - x_{i1} + x_{j3} - M\big(2 - y_{i1} - y_{j3}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{18}$$

$$T_{ij} \leq 2W + 3c + 2e + D - x_{i1} + x_{j3} + M\big(2 - y_{i1} - y_{j3}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{19}$$

$$T_{ij} \geq 2W + 2c + 2e + D - x_{i1} - x_{j4} - M\big(2 - y_{i1} - y_{j4}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{20}$$

$$T_{ij} \leq 2W + 2c + 2e + D - x_{i1} - x_{j4} + M\big(2 - y_{i1} - y_{j4}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{21}$$

$$T_{ij} \geq 2W + 4c + 2e + D + x_{i2} + x_{j3} - M\big(2 - y_{i2} - y_{j3}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{22}$$

$$T_{ij} \leq 2W + 4c + 2e + D + x_{i2} + x_{j3} + M\big(2 - y_{i2} - y_{j3}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{23}$$

$$T_{ij} \geq 2W + 3c + 2e + D + x_{i2} - x_{j4} - M\big(2 - y_{i2} - y_{j4}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{24}$$

$$T_{ij} \leq 2W + 3c + 2e + D + x_{i2} - x_{j4} + M\big(2 - y_{i2} - y_{j4}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{25}$$

$$T_{ij} \geq 4W + 3c + 2e + 2D - x_{i3} - x_{j4} - M\big(2 - y_{i3} - y_{j4}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{26}$$

$$T_{ij} \leq 4W + 3c + 2e + 2D - x_{i3} - x_{j4} + M\big(2 - y_{i3} - y_{j4}\big) \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{27}$$

$$T_{ji} = L - T_{ij} \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{28}$$

$$d'_2 \geq d_i - M(1 - y_{i2}) \quad \forall\ i \in I, \tag{29}$$

$$d'_3 \geq d_i - M(1 - y_{i3}) \quad \forall\ i \in I, \tag{30}$$

$$D = d'_2 + d'_3 + C, \tag{31}$$

$$L = 4W + 2D + 4c + 2e \tag{32}$$

$$z_{rij} + z_{rji} \leq y_{ir} \quad \forall\ i \in I^1, j \in I^2, r \in R, \tag{33}$$

$$z_{rij} + z_{rji} \leq y_{jr} \quad \forall\ i \in I^1, j \in I^2, r \in R, \tag{34}$$

$$z_{rij} + z_{rji} + 1 \geq y_{ir} + y_{jr} \quad \forall\ i \in I^1, j \in I^2, r \in R, \tag{35}$$

$$x_{ir} \geq 0, y_{ir} \in \{0, 1\} \quad \forall\ i \in I, r \in R, \tag{36}$$

$$z_{rij} \in \{0, 1\} \quad \forall\ i \in I, j \in \{I \setminus i\}, r \in R, \tag{37}$$

$$T_{ij} \geq 0 \quad \forall\ i \in I, j \in \{I \setminus i\}, \tag{38}$$

$$s_r \geq 0 \quad \forall\ r \in R, \tag{39}$$

$$A, W, D, L, d'_2, d'_3 \geq 0. \tag{40}$$

The objective function (1) consists of objectives (2) and (3), which are to be minimized simultaneously. Objective (2) seeks to minimize the material handling cost, while the value of objective (3) is defined as the area of the smallest rectangle enclosing all machines and the track in the resulting layout. Constraints (4)–(6) determine lower bounds on the width W and the layout area consumed by the machines located in each row. The objective (3) serves to make constraints (4)–(6) binding. Constraint (7) calculates the total area of the resulting layout. Constraints (8) and (9) ensure that each machine is placed in exactly one row. Constraints (10) and (11) guarantee that the minimum clearance between any two machines is satisfied, where $I^1 = \{1, \ldots, m-1\}$ and $I^2 = \{i+1, \ldots, m\}$ for all $i \in I^1$. The constant M is a sufficiently large value.

Constraints (12)–(28) calculate the travel distance between any two machines located in the same row or two different rows. Since the material is transferred by a unidirectional track, T_{ij} does not equal T_{ji}. Constraints (29) and (30) provide the lower bound on the depth of the deepest machine in rows 2 and 3, respectively. Constraints (31) and (32) determine the

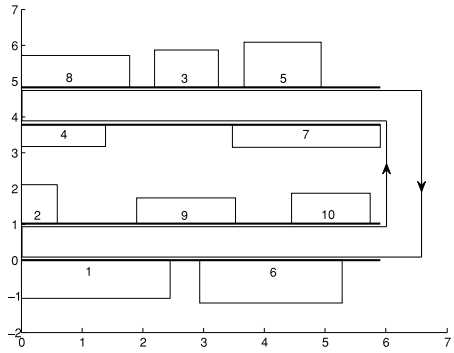

Fig. 2. An area-minimizing solution, s_a, produced by separating adjacent machines by their minimum allowable clearances.

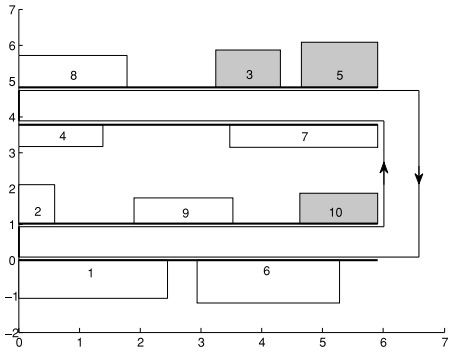

Fig. 3. The cost-minimizing solution, s_c, produced by linear programming. This is also an area-minimizing solution.

width of the bay pitch and the total length of the unidirectional track, respectively. Constraints (33)–(35) relate binary decision variables z_{rij} and y_{ir}. Finally, (36)–(40) define the decision variables.

A. Analysis of Structural Properties

A solution to the DBLP (i.e., a layout) captures the sequence of machines in each row as well as the exact location of each machine. For a given machine sequence in each row, there are an infinite number of solutions that may be obtained by adjusting the precise location of each machine (in continuous space). Fortunately, there is a single solution *for a given machine sequence* that simultaneously minimizes the area consumed and the material handling cost.

For example, consider a 10-machine instance (problem P_{10}^1 in Section V). A machine sequence, S, in the four rows is [8 3 5; 4 7; 2 9 10; 1 6]. The *depth* of the layout for this fixed sequence is determined by the deepest machine assigned to each row. Thus, the total area is minimized if the *width* of the layout corresponding to S is minimized. Finding the exact location of each machine that minimizes the area consumed is simply a matter of placing the machines with a separation between each pair of machines equal to their minimum allowable clearances. We denote this area-minimizing layout for this given sequence by s_a (see Figure 2). The other objective is to minimize material handling cost. Recall that the binary decision variables y_{ir} and z_{rij} specify the relative ordering of machines within each row. For a given sequence, S, the values of these decision variables may be readily determined. Thus, with these binary variables fixed, the mixed integer programming formulation becomes a pure linear program. Solving this LP, with only the objective of minimizing cost, yields the optimal values of continuous decision variables x_{ir} (i.e., the exact location of each machine). We denote this cost-minimizing solution as s_c (see Figure 3).

Compared to s_a (Figure 2), some machines in s_c (Figure 3) have shifted to the right (machines 3, 5, and 10, shaded in gray). This behavior shows that it may be advantageous for some machines to be separated by more than their minimum clearance requirements to minimize material handling costs. However, the widths of s_a and s_c are necessarily identical; in the DBLP, where material is moved via a unidirectional track, a solution with a larger width (area) will increase the length of the track. This would result in an increase in material handling costs. Thus, s_a and s_c consume the same (minimum) area.

Since s_c is the solution with both the minimum area and the minimum cost among the infinitely-many solutions associated with a given sequence, S, it must be the only non-dominated solution for S. The proposed solution approach exploits this property to efficiently explore the solution space.

IV. Solution Approach

The DBLP can be decomposed into a combinatorial subproblem to determine the assignment and sequence of machines, and a continuous subproblem to identify the optimal exact location of machines for the assignment and sequence.

It is well known that the machine assignment and sequence problem is NP-hard [20] and, as such, its computational complexity precludes the use of any exact approaches for problems of practical size. Instead, we turn to evolutionary algorithms (EAs), which are well-suited for NP-hard problems [21]. In this paper, a hybrid algorithm (Algorithm 1) is proposed that combines a *multi-objective genetic algorithm* (MOGA) with an LP.

A. A Multi-Objective Genetic Algorithm for the DBLP

The non-dominated sorting genetic algorithm II (NSGA-II) [22] is one of the most popular and effective multi-objective EA. We adapt the NSGA-II in three ways to more effectively find the set of non-dominated machine sequences. First, the DBLP requires a new solution encoding and appropriate genetic operators designed specifically for this mixed variable problem. Thus, we devise a concise solution encoding scheme as well as the crossover and mutation operators for the DBLP. Second, to improve the solution quality, a novel restart method is integrated within the NSGA-II framework to improve algorithm stability and produce additional non-dominated machine sequences. Finally, an archive set is introduced to store the best non-dominated solutions found so far, avoiding the loss of elitist solutions due to the restart.

1) Solution Encoding: Each individual represents a sequence of machines, S, across all four rows. An individual is encoded as two integer vectors: (1) a sequence vector representing a permutation of all machines; and (2) a breakpoint vector identifying the number of machines in each row.

Algorithm 1 Pseudocode for the MOGA-LP

```
% MOGA:
Produce n individuals to populate P_0;
A = ∅; % Start with an empty archive set
g = 0; % Initialize number of generations
while (g ≤ maxGen) do
    Evaluate each individual in P_g;
    Rank all individuals in P_g;
    Form population P'_g from P_g via selection operator;
    Form population P''_g via crossover operator on P'_g;
    Form population Q_g via mutation operator on P''_g;
    Evaluate each individual in Q_g;
    for (each non-dominated individual, S, in Q_g) do
        if (S is dominated by any individual in A) then
            Discard individual S (no update);
        else
            if (S dominates any individual in A) then
                Remove dominated individuals from A;
                Add S to A;
            else
                if (A is not full) then
                    Add S to A
                end if
            end if
        end if
    end for
    if (no updates to A for Iter generations) then
        Re-initialize the population as P_{g+1};
    else
        R_g ← P_g ∪ Q_g, and rank all individuals in R_g;
        P_{g+1} ← Top n individuals of R_g;
    end if
    g ← g + 1;
end while
% LP:
for all (S ∈ A) do
    Fix the values of y_ir and z_rij;
    Solve the resulting LP;
end for
Report all non-dominated solutions;
```

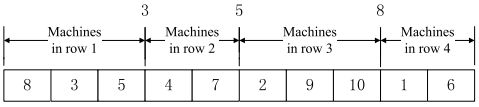

Fig. 4. An encoding of the sequence vector with breakpoint vector [3 5 8].

For example, the coding of a 10-machine sequence is given in Figure 4. The breakpoint vector, [3 5 8], indicates that there are 3 machines in row 1, $5-3=2$ machines in row 2, $8-5=3$ machines in row 3, and $10-8=2$ machines in row 4.

2) Population Initialization: Each individual in the initial population is produced as follows. First, the sequence vector is constructed as a random permutation of all machines. The breakpoint vector is given by the integer vector $[b_1\ b_2\ b_3]$, where b_1 is set to a random integer in the closed interval $[0, m]$ (m is the number of machines), $b_2 \in [b_1, m]$, and $b_3 \in [b_2, m]$.

3) Evaluation: Each individual corresponds to an infinite number of solutions, but possesses a single solution, s_c, that simultaneously minimizes the area and cost for this particular individual (machine sequence). While s_c may be easily obtained via the LP, repeatedly solving this LP during the search of MOGA is computationally expensive.

Recall that solution s_a can be readily constructed for a given individual S. Thus, the cost and area values of s_a can be calculated directly from the problem parameters. Solutions s_a and s_c both consume the minimum area for a given sequence, although the total cost for s_a may exceed that for s_c. We have found that the cost differences between these two solutions is small for a given sequence. Furthermore, for a sequence, if its s_a has a smaller cost value, then the cost value of its s_c tends to be smaller. We use the quickly-determined solution of s_a as a surrogate for s_c in evaluating a particular individual.

Based on their objective function values, all individuals in P_g are sorted by the well-known *fast non-dominated sorting approach* and *crowding-distance* [22] to assign each individual a ranking value.

4) Selection and Crossover: In generation g, a binary tournament selection (see [22]) is applied to population P_g to form a diversified population P'_g with n individuals. That is, each individual in P'_g is chosen as the better solution (with a smaller rank value) among two solutions randomly picked from population P_g. Next, a crossover operation is applied to pairs of individuals in P'_g to form population P''_g.

For each pair of individuals, the probability that this pair will undergo a crossover is given by $0 \le p_c \le 1$. The crossover operation is applied to both the sequence and breakpoint vectors. Thus, two parents (e.g., parents 1 and 2) will generate two children (e.g., child 1 and child 2).

For the sequence vector, uniform crossover [23] is adopted, where a binary string of length m is generated for each chosen pair of parents. To create child 1, genes of parent 1 corresponding to a "1" in the binary string are copied to the corresponding positions of the child. The remaining (missing) genes in child 1 are taken from parent 2, according to the order in which they appear in parent 2's sequence vector. Similarly, child 2 is created by selecting the genes of parent 2 corresponding to a "1" in the binary string, with the remaining genes copied from parent 1. For the breakpoint vector, each gene of a child is a random integer sampled from the values of the corresponding genes of the two parents.

5) Mutation: Each individual in population P''_g (with n individuals) undergoes a mutation operation with mutation probability $0 \le p_m \le 1$. As a result, an offspring population Q_g of size n is created. If an individual mutates, the mutation operator is applied to either its sequence vector or breakpoint vector (not both). Let p_m^s and p_m^b represent the probabilities that the sequence and breakpoint vectors, respectively, are mutated, where $p_m^s + p_m^b = 1$.

If the sequence vector is chosen, the mutation operator swaps the positions of two randomly-selected machines. If the mutation operator is applied to the breakpoint vector, it randomly selects one gene and sets the gene's value to a random

integer in a given range. To ensure the feasibility of the resulting breakpoint vector, the range is set to be $[0, b_2]$ ($[b_1, b_3]$, $[b_2, m]$) for the first (second, third) gene of the vector, where b_1 (b_2, b_3) is the value of the first (second, third) gene.

6) Create Population for Next Generation: If the archive set, A, is not updated for *Iter* consecutive generations, then the population is reinitialized. Otherwise, combine P_g and Q_g to form a population R_g of size $2n$. Individuals in R_g are then sorted by the fast non-dominated sorting approach and crowding-distance. The top n individuals in R_g are selected to form the population of the next generation, P_{g+1}.

B. Improving Solutions via Linear Programming

After the MOGA is terminated, archive set A contains a collection of non-dominated individuals, each of which represents a machine sequence. Although an arbitrary machine sequence, S, corresponds to an infinite number of solutions to a DBLP, there is only one non-dominated solution, s_c, for S. This solution is identified by solving an LP to find the true optimal location for each machine in the particular sequence, S. Thus, the final Pareto solutions for the DBLP may be found by solving the LP for each machine sequence in the archive set, keeping only those non-dominated solutions.

V. Experiments

The proposed MOGA-LP is applied to representative problem instances and compared against an exact approach (CPLEX) to evaluate its effectiveness.

DBLP instances were generated as follows. First, the weighted material flow values, f_{ij}, were constructed by assuming that 8 to 10 product types are to be produced, each with a production quantity of 20 to 50 units. Between 25% and 75% of the machines are visited by a particular product type, with each product being assigned a sequence of visits to those machines. Thus, f_{ij} was calculated as the sum of products whose routes include machine i immediately preceding machine j. Parameters c and e are $\sim$unif[0.5, 2.0]; d_i is $\sim$unif[0.5, 1.5]; w_i is $\sim$unif[0.5, 2.5]; a_{ij} is $\sim$unif[0.25, 2.25]; and C is $\sim$unif[$\min_{i\in I}\{d_i\}$, $\max_{i\in I}\{d_i\} + 0.5$].

The problem sizes of the DBLP are chosen to be 10, 20, 30, 50 and 75 machines. Three representative problem instances are generated for each problem size. Instances P^1_{10}, P^2_{10} and P^3_{10} contain 10 machines. Similarly, instances $P^1_{20} - P^3_{20}$, $P^1_{30} - P^3_{30}$, $P^1_{50} - P^3_{50}$, and $P^1_{75} - P^3_{75}$ contain 20, 30, 50 and 75 machines, respectively.

A. Algorithm Parameters

The mutation operator for the sequence vector actually preforms a local search. It well knows that local search is effective for sequence problems. As such, we set $p_m = 1$ to allow the mutation operator to be applied to each individual. The probabilities of applying the mutation to either the sequence or breakpoint vectors are given by $p^s_m = 0.8$ and $p^b_m = 0.2$, respectively. We let the crossover probability $p_c = 0.2$ and the population size $n = 100$. To make the MOGA adequately converge, the number of generations, *maxGen*, is set to 150,000, 200,000, 250,000, 300,000 and 350,000 for problem instances with 10, 20, 30, 50 and 75 machines, respectively. We set the size of archive set A to be 50 for all problem instances. *Iter* is chosen as 1000, 1000, 1500, 2000, and 3000 for instances with 10, 20, 30, 50 and 75 machines, respectively.

B. Comparison With an Exact Approach

Since the DBLP is newly formulated in this paper, there exist no solution approaches for comparison. A commercial mathematical programming solver (IBM ILOG CPLEX version 12.5.1.0) is used to solve the mixed integer programming formulation. The two objectives are combined to form a single objective. To make the two objectives have approximately the same magnitude, we normalize them via the ideal objective vector, $z^* = (z^*_1, z^*_2)$ [24]. The resulting linearly-weighted objective function is given by

$$\text{Minimize } \alpha \frac{Obj_1}{z^*_1} + (1-\alpha)\frac{Obj_2}{z^*_2}, \tag{41}$$

where z^*_1 (z^*_2) is the ideal objective value, i.e., the optimal objective value produced by using CPLEX to solve the formulation with the single objective of cost (area). To produce a set of Pareto solutions, α is increased from 0 to 1 by a small step size (0.1); CPLEX is used to solve the resulting problem for each value of α.

For P^1_{10}–P^3_{10}, CPLEX is able to find the true optimal solution for each value of α. CPLEX cannot produce optimal solutions for other instances in a reasonable time. Thus, the runtime of CPLEX is restricted to one hour for each α value when solving P^1_{20}–P^3_{20} and P^1_{30}–P^3_{30}. CPLEX cannot effectively handle P^1_{50}–P^3_{50} and P^1_{75}–P^3_{75}, so it is not applied to those problem instances.

C. Experimental Results

The MOGA-LP was programmed in C and was executed on an HP 8100 Elite desktop PC with a quad-core Intel i7-4770 3.4GHz CPU, 16G memory and Ubuntu Linux 10.10 operating system. Five independent runs of MOGA-LP were performed for each problem instance.

We first show the Pareto solutions produced by the MOGA-LP and CPLEX approaches. The non-dominated machine sequences obtained by the 5 runs of MOGA (denoted by $\cdot$, $\circ$, $+$, $\times$ and $\triangledown$, respectively) for P^1_{10} are presented in Figure 5. The five runs obtain the same five machine sequences, where S is used as an example in Section III and is illustrated in Figure 2. For each machine sequence identified by MOGA, the LP was used to create a solution. Thus, five solutions (s_c and s_1-s_4) are produced and are represented by ☆. Solution s_c corresponding to the sequence S is shown in Figure 3. Recall that each individual (sequence) is evaluated in the MOGA by the solution constructed by separating adjacent machines by their minimum required clearance (namely s_a). However, the minimum-cost solution, s_c, has the same area as s_a, but has a cost that cannot exceed that of s_a. Hence, s_c will frequently appear directly to the left of the machine sequences stored in the archive set. Removing those dominated solutions

TABLE II
COMPARISON OF HV AND C-METRIC FOR MOGA-LP AND CPLEX

	MOGA-LP			CPLEX		
Instance	Mean # of Solutions	Mean HV	Mean C(A, B) (%)	# of Solutions	HV	Mean C(B, A) (%)
P_{10}^1	**3.0**	**414.61**	**0**	2	0.00	0.00
P_{10}^2	**6.0**	**27,065.94**	**0**	4	26,808.45	0.00
P_{10}^3	**5.0**	**11,508.51**	**0**	4	10,885.63	0.00
P_{20}^1	**11.4**	**256,215.59**	**0**	3	135,608.85	100.00
P_{20}^2	**11.8**	**252,151.70**	**0**	6	121,951.41	96.67
P_{20}^3	**13.4**	**594,508.95**	**0**	5	333,283.72	100.00
P_{30}^1	**15.6**	**2,645,883.62**	**0**	3	686,502.17	100.00
P_{30}^2	**16.2**	**1,550,845.71**	**0**	3	494,236.63	100.00
P_{30}^3	**14.4**	**1,283,274.44**	**0**	2	82,877.35	100.00

A and B correspond to CPLEX and MOGA-LP, respectively. The better results are highlighted in bold font.

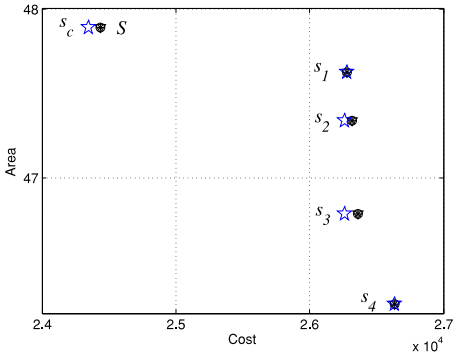

Fig. 5. Non-dominated machine sequences and their corresponding solutions found by MOGA-LP for P_{10}^1.

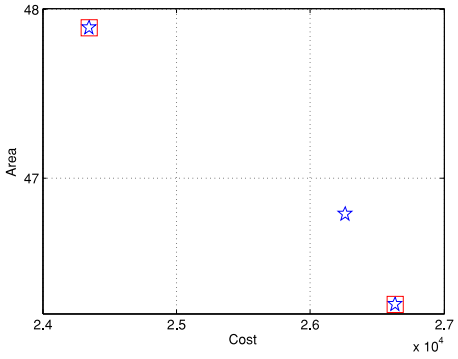

Fig. 6. Pareto solutions found by MOGA-LP and CPLEX for P_{10}^1.

(s_1 and s_2), the three final Pareto solutions are represented by ☆ in Figure 6. CPLEX produces two Pareto solutions (represented by large □), which are also captured by MOGA-LP. Thus, MOGA-LP is able to find the true Pareto front for P_{10}^1. MOGA-LP can also identify the true Pareto optimal solutions for P_{10}^2 and P_{10}^3.

For the sake of brevity, we use P_{30}^1 as a sample of other problem instances to show its Pareto solutions, as presented in Figure 7. Pareto solutions produced in the five runs of MOGA-LP are denoted by ·, ∘, +, × and ▽, respectively; the Pareto solutions of CPLEX are represented by large □. Compared to CPLEX, MOGA-LP is able to find more Pareto solutions distributed in a wide range, and the solution quality of MOGA-LP is clearly better than that of CPLEX.

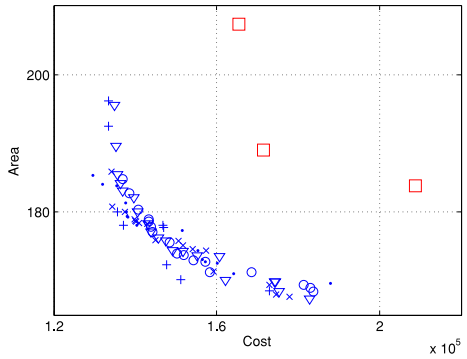

Fig. 7. Pareto solutions found by MOGA-LP and CPLEX for P_{30}^1.

Two commonly used metrics for multiobjective optimization algorithm, hypervolume (HV) and set coverage (C-metric) [25], are adopted to evaluate the algorithm's performance. In the objective space, let $y^* = (y_1^*, y_2^*)$ represent a point which is dominated by any Pareto solution in a set S. Then the HV value of S (with regard to y^*) is the volume of the region dominating y^* yet dominated by S. A larger HV value indicates better solution quality. Let A and B be two sets containing Pareto solutions. Define C-metric $C(A, B)$ as the percentage of the solutions in B that are dominated by as least one solution in A. The smaller the $C(A, B)$ value, the better the quality of B relative to A.

The HV and C-metric values of the Pareto solutions produced by CPLEX and MOGA-LP for each problem instance are presented in Table II. The y_i^* $(i = 1, 2)$ in HV is set to be the greatest value of the ith objective among all the Pareto solutions created by the two approaches. The mean HV is the mean value of the HVs of five Pareto solution sets generated by MOGA-LP. The mean C-metric is computed by comparing each pair of Pareto solution sets from the two approaches. The mean number of solutions is also given in Table II.

The C-metric values of CPLEX and MOGA-LP for P_{10}^1–P_{10}^3 are all zero since they produce the coincident solutions for the three instances. For other instances, MOGA-LP performs significantly better than CPLEX, in terms of both the HV and C-metric values, indicating that MOGA-LP outperforms CPLEX in both aspects of convergence and distribution. In addition, MOGA-LP generates more Pareto solutions than CPLEX for each problem instance.

TABLE III
COMPARISON OF COMPUTATIONAL TIME (IN SECONDS) FOR MOGA-LP AND CPLEX

Instance	MOGA-LP			CPLEX Time
	MOGA Mean Time	LP Mean Time	Total Mean Time	
P_{10}^1	150	0.4	150.4	3019
P_{10}^2	151	0.5	151.5	2641
P_{10}^3	151	0.4	151.4	3479
P_{20}^1	319	3.5	322.5	39600
P_{20}^2	318	4.2	322.2	39600
P_{20}^3	319	4.2	323.2	39600
P_{30}^1	615	10.1	625.1	39600
P_{30}^2	617	10.2	627.2	39600
P_{30}^3	613	9.4	622.4	39600
P_{50}^1	1579	25.9	1604.9	–
P_{50}^2	1581	30.5	1611.5	–
P_{50}^3	1579	21.1	1600.1	–
P_{75}^1	3846	52.0	3898.0	–
P_{75}^2	3819	53.5	3872.5	–
P_{75}^3	3842	62.6	3904.6	–

TABLE IV
MEAN COMPUTATIONAL TIME OF MOGA PER GENERATION OF EVOLUTION

Instances	P_{10}^1–P_{10}^3	P_{20}^1–P_{20}^3	P_{30}^1–P_{30}^3	P_{50}^1–P_{50}^3	P_{75}^1–P_{75}^3
Mean Time (s)	0.0010	0.0016	0.0025	0.0053	0.0110

TABLE V
COMPARISON OF THE MODIFIED NSGA-II AND THE NSGA-II WITHOUT REINITIALIZATION FOR P_{10}^1 AND P_{30}^2

Inst	The modified NSGA-II				The NSGA-II without reinitialization			
	Mean # of Seqs	Mean HV	Mean Time (s)	Mean C(A,B)(%)	Mean # of Seqs	Mean HV	Mean Time (s)	Mean C(B,A)(%)
P_{10}^1	**5.0**	**8,434.23**	150.4	**0.00**	2.6	3,538.60	**140.2**	70.00
P_{30}^2	**16.2**	**1,259,631.24**	627.2	**9.97**	10.8	954,814.17	**595.2**	70.95

A and B correspond to the NSGA-II without reinitialization and the modified NSGA-II, respectively. The better results are highlighted in bold font.

The computational time of MOGA-LP and CPLEX is presented in Table III. MOGA-LP takes much less time than the exact approach (CPLEX). For each problem size, MOGA consumes similar computational time for each of its three instances while the mean runtime for the LP phase may be much different for each instance. The LP is run for each non-dominated sequence created by MOGA, so that the mean LP runtime for a problem instance is proportional to the number of non-dominated sequences created by MOGA for this instance.

Table III indicates that the computational time of MOGA-LP is mainly consumed by MOGA. Table IV shows the mean CPU time per generation of MOGA. Not surprisingly, the computational time increases with problem size. We used a well-known statistical software package, SPSS, to perform regression analysis on the relationship between problem size (x) and the computational time (y). Their relationship is well fitted by the following quadratic function:

$$y = \left(1.809 \times 10^{-6}\right)x^2 - \left(6.643 \times 10^{-7}\right)x + 0.001. \quad (42)$$

The regression model's *coefficient of determination*, which is in [0, 1] and reflects *goodness of fit*, equals 1, and its *standard error of the estimate* is approximately 0. The relationship between x and y is polynomial, not exponential, such that the increase in time is quite manageable.

To investigate the effect of reinitialization, the reinitialization and archive set are removed from the modified NSGA-II. In the interest of brevity, only two representative problem instances (P_{10}^1 and P_{30}^2) are shown, although the other problem instances have similar results. Both algorithms were run 150,000 and 250,000 generations for P_{10}^1 and P_{30}^2, respectively. As summarized in Table V, the modified NSGA-II outperforms the NSGA-II without reinitialization, in terms of both HV and C-metric.

VI. CONCLUSION

This paper defines a double-bay layout problem arising in the semiconductor manufacturing context. The DBLP considers the assignment and order of machines in each bay, as well as the exact location of each machine, and involves two objectives of material handling cost and layout area. A mixed integer programming model of the DBLP is established. A novel hybrid approach is proposed that combines a multi-objective meta-heuristic and a mathematical model-based approach. A modified NSGA-II is used to find a set of non-dominated machine sequences, and the LP determines the optimal locations of machines in each non-dominated sequence, thus creating a collection of final Pareto solutions. Experiments show that the proposed MOGA-LP effectively solves this problem and has better performance than either an exact approach or the traditional NSGA-II. By considering both space and material handling costs simultaneously, a compact layout that shortens component travel distances is produced. This reduces both one time (footprint) costs and recurring (operational) costs in semiconductor manufacturing.

Future research directions include the generalization of the DBLP to more than two bays. Another relevant extension is to consider bi-directional tracks for material transfer.

REFERENCES

[1] T. Yang, M. Rajasekharan, and B. A. Peters, "Semiconductor fabrication facility design using a hybrid search methodology," *Comput. Ind. Eng.*, vol. 36, no. 3, pp. 565–583, 1999.

[2] B. A. Peters and T. Yang, "Integrated facility layout and material handling system design in semiconductor fabrication facilities," *IEEE Trans. Semicond. Manuf.*, vol. 10, no. 3, pp. 360–369, Aug. 1997.

[3] R. D. Meller, "The multi-bay manufacturing facility layout problem," *Int. J. Prod. Res.*, vol. 35, no. 5, pp. 1229–1237, 1997.

[4] J. Chae and B. A. Peters, "Layout design of multi-bay facilities with limited bay flexibility," *J. Manuf. Syst.*, vol. 25, no. 1, pp. 1–11, 2006.

[5] J. L. Turley, *The Essential Guide to Semiconductors*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[6] T. Yang and B. A. Peters, "A spine layout design method for semiconductor fabrication facilities containing automated material handling systems," *Int. J. Oper. Prod. Manag.*, vol. 17, no. 5, pp. 490–501, 1997.

[7] D. Nazzal and A. El-Nashar, "Survey of research in modeling conveyor-based automated material handling systems in wafer fabs," in *Proc. Winter Simulat. Conf.*, Washington, DC, USA, 2007, pp. 1781–1788.

[8] I. Castillo and B. A. Peters, "Integrating design and production planning considerations in multi-bay manufacturing facility layout," *Eur. J. Oper. Res.*, vol. 157, no. 3, pp. 671–687, 2004.

[9] J. H. Ting and J. M. A. Tanchoco, "Unidirectional circular layout for overhead material handling systems," *Int. J. Prod. Res.*, vol. 38, no. 16, pp. 3913–3935, 2000.

[10] S. S. Heragu and A. Kusiak, "Machine layout problem in flexible manufacturing systems," *Oper. Res.*, vol. 36, no. 2, pp. 258–268, Mar./Apr. 1988.

[11] J. Chung and J. M. A. Tanchoco, "The double row layout problem," *Int. J. Prod. Res.*, vol. 48, no. 3, pp. 709–727, 2010.

[12] Z. Zhang and C. C. Murray, "A corrected formulation for the double row layout problem," *Int. J. Prod. Res.*, vol. 50, no. 15, pp. 4220–4223, 2012.

[13] C. C. Murray, A. E. Smith, and Z. Q. Zhang, "An efficient local search heuristic for the double row layout problem with asymmetric material flow," *Int. J. Prod. Res.*, vol. 51, no. 20, pp. 6129–6139, 2013.

[14] C. C. Murray, X. Q. Zuo, and A. E. Smith, "An extended double row layout problem," in *Proc. 12th Int. Mater. Handling Res. Colloquium*, Gardanne, France, 2012, pp. 554–569.

[15] X. Q. Zuo, C. C. Murray, and A. E. Smith, "Solving an extended double row layout problem using multiobjective tabu search and linear programming," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1122–1132, Oct. 2014.

[16] H. Ahonen, A. G. de Alvarenga, and A. R. S. Amaral, "Simulated annealing and tabu search approaches for the corridor allocation problem," *Eur. J. Oper. Res.*, vol. 232, no. 1, pp. 221–233, 2014.

[17] S. Kulturel-Konak and A. Konak, "A new relaxed flexible bay structure representation and particle swarm optimization for the unequal area facility layout problem," *Eng. Optim.*, vol. 43, no. 12, pp. 1263–1287, 2011.

[18] M. Saravanan and S. G. Kumar, "Different approaches for the loop layout problems: A review," *Int. J. Adv. Manuf. Technol.*, vol. 69, nos. 9–12, pp. 2513–2529, 2013.

[19] F. Ozcelik and A. A. Isliera, "Generalisation of unidirectional loop layout problem and solution by a genetic algorithm," *Int. J. Prod. Res.*, vol. 49, no. 3, pp. 747–764, 2011.

[20] J. R. Montoya-Torres, "A literature survey on the design approaches and operational issues of automated wafer-transport systems for wafer fabs," *Prod. Plan. Control*, vol. 17, no. 7, pp. 648–663, 2006.

[21] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Rel. Eng. Syst. Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[23] M. Kaya, "The effects of two new crossover operators on genetic algorithm performance," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 881–890, 2011.

[24] K. Miettinen, *Nonlinear Multiobjective Optimization*. Boston, MA, USA: Kluwer, 1999.

[25] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

Xingquan Zuo received the Ph.D. degree in control theory and control engineering from the Harbin Institute of Technology, Harbin, China, in 2004. He is currently an Associate Professor with the School of Computer Science, Beijing University of Posts and Telecommunications. From 2004 to 2006, he was a Post-Doctoral Research Fellow with the Automation Department of Tsinghua University. From 2012 to 2013, he was a Visiting Scholar with the Department of Industrial and Systems Engineering, Auburn University, AL, USA. He has published over 70 research papers in journals and conferences, two books, and several book chapters. His research interests are in system optimization and scheduling, evolutionary computation, data mining with applications and intelligent transportation systems. He is a member of ACM and the IEEE Computational Intelligence Society, a Committee Member of the Natural Computation Society of Chinese Association for Artificial Intelligence and Transportation Model and Simulation Society of Chinese Association for System Simulation, and a Senior Member of the Chinese Association for Artificial Intelligence. He served in program committee of many conferences.

Chase C. Murray received the B.S. and M.Eng. degrees in industrial engineering from Texas A&M University, College Station, TX, USA, in 1998 and 2000, respectively, and the Ph.D. degree in industrial and systems engineering from the University at Buffalo, Buffalo, NY, USA, in 2010. He was an Industrial Engineer in the semiconductor industry with Intel and Dallas Semiconductor. He is currently an Assistant Professor in industrial and systems engineering with the University at Buffalo. His research interests are in the area of operations research, with a particular focus on military applications of vehicle routing.

Alice E. Smith is the Joe W. Forehand/Accenture Distinguished Professor of the Industrial and Systems Engineering Department with Auburn University, where she served as the Department Chair from 1999 to 2011. She also has a joint appointment with the Department of Computer Science and Software Engineering. Her research focus is analysis, modeling, and optimization of complex systems with emphasis on computation inspired by natural systems. She holds one U.S. patent and several international patents and has authored over 200 publications. She has over 2600 ISI Web of Science citations with an H-index of 24 and over 8000 Google Scholar citations with an H-index of 41. Several of her papers are among the most highly cited in their respective journals, including the most cited paper of *Reliability Engineering & System Safety* and the second most cited paper of the IEEE TRANSACTIONS ON RELIABILITY. She is an Area Editor of both *INFORMS Journal on Computing* and *Computers & Operations Research* and an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. She has been a Principal Investigator on over $7.5 million of sponsored research with funding by NASA, U.S. Department of Defense, Missile Defense Agency, National Security Agency, NIST, U.S. Department of Transportation, Lockheed Martin, Adtranz (currently Bombardier Transportation), the Ben Franklin Technology Center of Western Pennsylvania, and the U.S. National Science Foundation, from which she has been awarded 16 grants, including a CAREER grant in 1995 and an ADVANCE Leadership grant in 2001. She was a Fulbright Senior Scholar with Bilkent University, Turkey, in 2013. She is a Fulbright Specialist with EAFIT, Medellin, Colombia, in 2016, and she will be a Senior Fulbright Fellow with Pontifical Catholic University of Valparaíso, in 2017.