Journal of Scheduling **8:** 387–426, 2005.
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

SEQUENCING AND SCHEDULING IN ROBOTIC CELLS: RECENT DEVELOPMENTS

MILIND DAWANDE[1], H. NEIL GEISMAR[2], SURESH P. SETHI[1], AND CHELLIAH SRISKANDARAJAH[1]

[1] *University of Texas at Dallas, School of Management, P.O. Box 830688 SM 30, Richardson, TX 75083-0688, USA*
[2] *College of Business, Prairie View A & M University, P.O. Box 519, Prairie View, TX 77446-0519*

ABSTRACT

A great deal of work has been done to analyze the problem of robot move sequencing and part scheduling in robotic flowshop cells. We examine the recent developments in this literature. A robotic flowshop cell consists of a number of processing stages served by one or more robots. Each stage has one or more machines that perform that stage's processing. Types of robotic cells are differentiated from one another by certain characteristics, including robot type, robot travel-time, number of robots, types of parts processed, and use of parallel machines within stages. We focus on cyclic production of parts. A cycle is specified by a repeatable sequence of robot moves designed to transfer a set of parts between the machines for their processing.

We start by providing a classification scheme for robotic cell scheduling problems that is based on three characteristics: machine environment, processing restrictions, and objective function, and discuss the influence of these characteristics on the methods of analysis employed. In addition to reporting recent results on classical robotic cell scheduling problems, we include results on robotic cells with advanced features such as dual gripper robots, parallel machines, and multiple robots. Next, we examine implementation issues that have been addressed in the practice-oriented literature and detail the optimal policies to use under various combinations of conditions. We conclude by describing some important open problems in the field.

KEY WORDS: manufacturing, robotic cell, cyclic solutions, flexible manufacturing

1. INTRODUCTION

Modern manufacturing processes have incorporated automation and repetitive processing. Employing their efficiency is necessary to compete in the marketplace. Many industries use computer-controlled material handling systems to convey raw materials through the multiple processing stages required to produce a finished product or part. In order to use such systems in a manner that provides maximum return on investment, efficient sequences of actions and schedules of parts must be found.

This study focuses on sequencing and scheduling for a particular type of automated material handling system in cellular manufacturing: robotic cells. Robotic cells consist of an input device, a series of processing stages, each of which performs a different process on each part in a fixed sequence, an output device, and robots that transport the parts within the cell. (Unless otherwise noted, all cells considered herein have one robot.) Each stage has one or more machines that perform that stage's processing. In essence, a robotic cell is a flowshop with blocking (Pinedo, 1995) that has common servers which perform all transfers of materials between processing stations. See Figure 1.

Correspondence to: M. Dawande, E-mail: milind@utdallas.edu

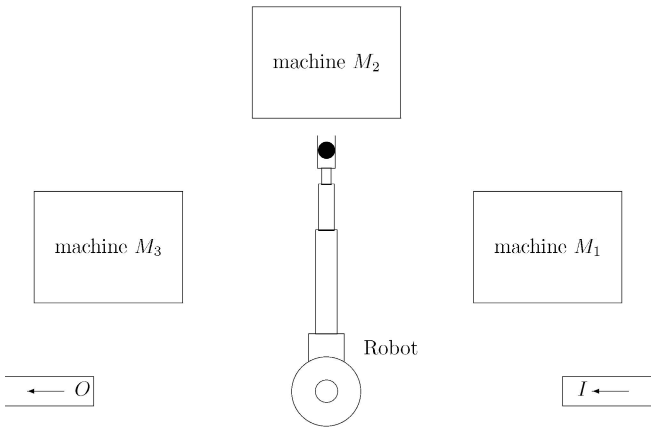

Figure 1. Three-machine robot centered cell.

Many diverse industries use robotic cells (see Section 11). A large number of studies address the semiconductor manufacturing industry: Akçali, Nemoto, and Uzsoy (2001), Kumar, Ramanan, and Sriskandarajah (2005), Perkinson et al. (1994), Perkinson, Gyurcsik, and McLarty (1996), Venkatesh et al. (1997), and Wood (1996). Other implementations for which studies have been published include machining of castings for truck differential assemblies (Asfahl, 1985), a single hoist electroplating line for printed circuit boards (Lei and Wang, 1994), testing and inspecting boards used in mainframe computers (Miller, 1984), crane scheduling for computer integrated manufacturing, textile mills, and engine block manufacturing (Su and Chen, 1996).

As manufacturers implement larger and more complex robotic cells, more sophisticated models and algorithms are required to optimize these operations. To meet this demand, there have been many studies. Some date as far back as the late 1970s, but the majority have been performed since 1990. Given the increasing importance of automated manufacturing, we provide a survey of the recent literature concerning the optimal operation of a robotic cell. Our strategy will be to build upon the survey of Crama et al. (2000) by providing a different perspective on some issues and by explaining the results that have been derived since that paper's publication. In addition to providing an overview of current results, we also discuss unresolved issues and possible future research directions.

The outline of this study is as follows: We begin with an historical overview of robotic cell scheduling publications (Section 2). Next, a classification scheme for robotic cell scheduling problems is presented in Section 3. In subsequent sections, we use this classification to specify the problems being discussed. Section 4 discusses relevant cell data whose values influence a cell's performance and defines some basic notation for subsequent use. Section 5 is devoted to cyclic production in robotic cells. We elaborate on cycles and define terms that are used to classify them. With an eye toward understanding the optimality of cycles, we discuss the computation of a cycle's execution time and present cycles that are optimal under certain cell data. Following that, we consider the more general multi-unit cycles and discuss some approximation results. Sections 6, 7, 8, and 9 examine the recent results for robotic cells with advanced features: multiple part-types, dual grippers, parallel machines, and multiple robots, respectively. After examining topics that have, to this point, only been addressed in the practice-oriented literature (Section 10), we discuss how robotic cells are used in practice (Section 11). Section 12 outlines some of the fundamental open problems for future study. We conclude in Section 13.

2. HISTORICAL OVERVIEW

In one of the early papers in the field for robotic cell sequencing, Bedini, Lisini, and Sterpos (1979) develop heuristic procedures for optimizing the working cycle of an industrial robot equipped with two independent arms. Baumann et al. (1981) derive models to determine robot and machine utilization. Maimon and Nof (1985) and Nof and Hannah (1989) study cells with multiple robots. Devedzic (1990) proposes a knowledge-based system to control the robot.

Wilhelm (1987) classifies the computational complexity of various scheduling problems in assembly cells. In a later study, Hall and Sriskandarajah (1996) survey scheduling problems with blocking and no-wait conditions and classify their computational complexities.

Early studies use simulation to compute cycle times. Kondoleon (1979) uses computer modeling to simulate robot motions in order to analyze the effects of configurations on the cycle time. Claybourne (1983) performs a simulation to study the effects that sequencing robot activities has on throughput. Asfahl (1985) simulates the actions of a robotic cell with three machines to demonstrate the transition from cold start to steady state cyclic operations. Błażewicz, Sethi, and Sriskandarajah (1989) develop an analytical method to derive cycle time formulas for robotic cells. Dixon and Hill (1990) compute cycle times by using a database language to simulate robotic cells.

Sethi et al. (1992) set the agenda for most subsequent studies on cells. They provide analytical solutions to the robot sequencing problem for two-machine and three-machine cells that produce identical parts, and for two-machine cells that produce different parts. Logendran and Sriskandarajah (1996) generalize this work to cells with different types of robots and with more general robot travel-times. Brauner and Finke (1997, 1999, 2001a, 2001b) perform several studies that compare 1-unit cycles with multi-unit cycles. Crama and van de Klundert (1997a) develop a polynomial algorithm for finding an optimal 1-unit cycle in an additive travel-time cell (travel-times are defined in Section 3.2.2). Dawande, Sriskandarajah, and Sethi (2002) do the same for constant travel-time cells. Brauner, Finke, and Kubiak (2003) prove that finding the optimal robot move sequence in a robotic cell with general travel-times, called a Euclidean robotic cell, is NP-hard. Hall, Kamoun, and Sriskandarajah (1997, 1998) and Sriskandarajah, Hall, and Kamoun (1998) study part scheduling problems and their complexities for cells that process parts of different types.

Research on cells with no-wait or interval pickup has been performed in parallel with the above-mentioned studies on free-pickup cells (see Section 3.2.1 for information on pickup characteristics). Levner, Kats, and Levit (1997) develop an algorithm that finds an optimal 1-unit cycle in a no-wait cell that produces identical parts. Agnetis (2000) finds optimal part schedules for no-wait cells with two or three machines. Agnetis and Pacciarelli (2000) study the complexity of the part scheduling problem for three-machine cells. Che, Chu, and Levner (2003) present a polynomial algorithm to find an optimal two-unit cycle in no-wait cells that produce identical parts or two part-types. Kats and Levner (2002) address no-wait cells with multiple robots.

An early work on interval robotic cells is by Lei and Wang (1994), who use a branch and bound search process. Chen, Chu, and Proth (1998) use branch and bound, linear programming, and bi-valued graphs to find optimal 1-unit cycles, and Che, Chu, and Chu (2002) employ these techniques to find optimal multi-unit cycles. Kats, Levner, and Meyzin (1999) solve this problem using a method similar to that used by Levner, Kats, and Levit (1997) for no-wait cells. Complexity results for such a system are presented in Crama (1997), Crama and van de Klundert (1997b), and van de Klundert (1996).

3. A CLASSIFICATION SCHEME

We now present a classification scheme for robotic cell scheduling problems. As in the classification scheme for classical scheduling problems (Graham et al., 1979), we distinguish problems based on three characteristics: machine environment (α), processing characteristics (β), and objective function (γ). A problem is then represented by the form $\alpha|\beta|\gamma$. Following the discussion of these characteristics, we detail our classification in Section 3.4 and also provide a pictorial representation in Figure 2.

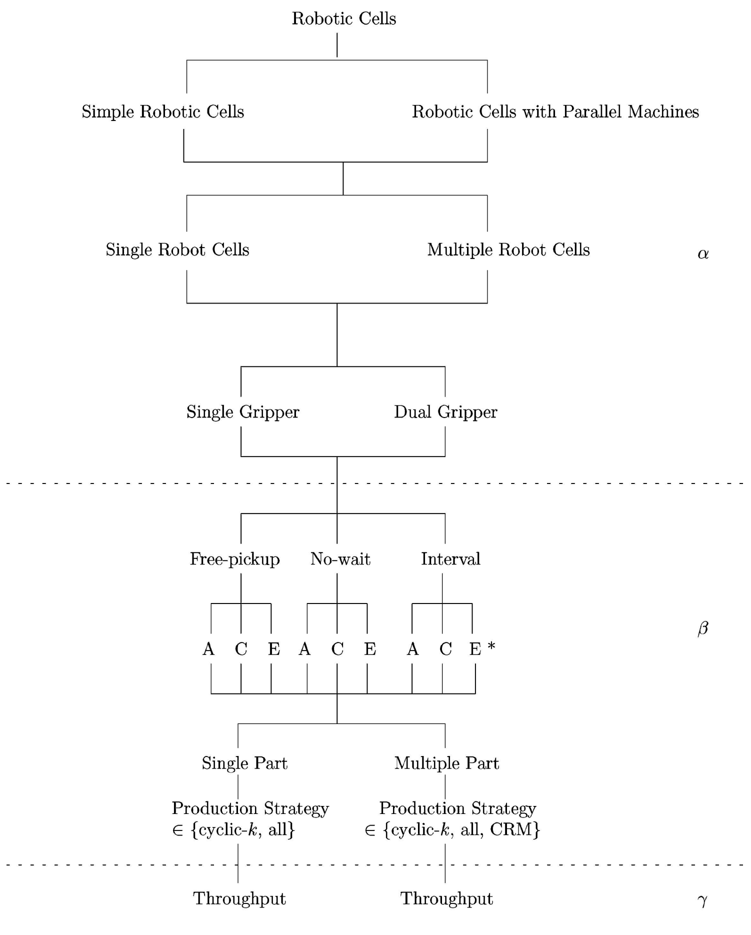

Figure 2. A classification for robotic flowshops.

3.1. Machine environment

We now describe characteristics that are represented in the first field of our classification scheme.

3.1.1. Number of machines per stage

If each processing stage has only one machine, the robotic cell is called a *simple robotic cell* or a *robotic flowshop*. Such cells contrast with *robotic cells with parallel machines*, in which at least one processing stage has more than one machine. Cells with parallel machines are discussed in Section 8.

A typical simple robotic cell contains m processing machines: $M_1, M_2, \ldots, M_m$. Let $M = \{1, 2, \ldots, m\}$ be the set of indices of these machines. The robot obtains a part from the input device (I, often denoted M_0), carries the part to the first machine (M_1), and loads the part. After M_1 completes its processing on the part, the robot unloads the part from M_1 and transports it to M_2, upon which it loads the part. This pattern continues for machines $M_3, M_4, \ldots, M_m$, where m represents the number of machines in the cell. After M_m has completed its processing on the part, the robot unloads the part and carries it to the output device (O, often denoted M_{m+1}). In some implementations, the input device and the output device are at the same location, and this unit is called a *load lock*. A three-machine simple robotic cell is depicted in Figure 1.

This description should not be misconstrued as implying that the robot remains with each part throughout its processing by each machine. Often, after loading a part onto a machine, the robot moves to another machine or to the input device in order to collect another part to transport it to its next destination. Determining which sequence of such moves is the optimal has been the focus of the majority of research on robotic cell sequencing and scheduling.

3.1.2. Number of robots

Manufacturers employ additional robots in a cell in order to increase throughput by increasing the material handling capacity. Cells with one (resp. more than one) robot are called *single* (resp. *multiple*) *robot cells*. Most studies in the literature analyze single robot cells. Multiple robot cells are discussed in Section 9.

3.1.3. Type of robot

A single gripper robot can hold only one part at a time. In contrast, dual gripper robots can hold two parts simultaneously. In a typical usage of this capability, the robot holds one part while the other gripper is empty; the empty gripper unloads a machine, the robot repositions the second gripper, and it loads that machine. Dual gripper robots are discussed in Section 7.

3.2. Processing characteristics

Four different processing characteristics are specified in the second field. We describe three in this section. The fourth, production strategy, is detailed in Section 3.4.

3.2.1. Pickup criterion

A significant feature of the robotic cells that we study is that they have no buffers for intermediate storage. All parts must be either in the input device, on one of the machines, in the output device, or on the robot.

Robotic cells can be partitioned into three types—*free-pickup, no-wait, interval*—based on the pickup criterion. Crama et al. (2000) refer to these three types as *unbounded processing windows,* *zero-width processing windows, processing windows,* respectively. For all three types, a part that has completed processing on M_i cannot be loaded onto M_{i+1} for its next processing unless M_{i+1} is unoccupied, $i = 0, \dots, m$. In free-pickup cells, this is the only pickup restriction: a completed part may remain on M_i indefinitely.

For the more restrictive no-wait cells, a part must be removed from machine $M_i, i \in M$, and transferred to machine M_{i+1} as soon as M_i completes processing that part. Such conditions are commonly seen in steel manufacturing or plastic molding, where the raw material must maintain a certain temperature, or in food canning to ensure freshness (Hall and Sriskandarajah, 1996). Results for no-wait cells can be found in Agnetis (2000), Agnetis and Pacciarelli (2000), Che, Chu, and Levner (2003), Hall and Sriskandarajah (1996), Kats and Levner (2002), and Levner, Kats, and Levit (1997).

In interval robotic cells, each stage has a specific interval of time—a processing time window—for which a part can be processed on that stage. This is applicable, for example, for the Hoist Scheduling Problem on an electroplating line (Che, Chu, and Chu, 2002; Chen, Chu, and Proth, 1998; Lei and Wang, 1994). Printed circuit boards are placed in a series of tanks with different solvents. Each tank has a specific interval of time—a processing window—for which a card can remain immersed.

Unless otherwise specified, all cells discussed herein have the free-pickup criterion.

3.2.2. *Travel-time metric*

The robot's travel-time between machines greatly influences a cell's performance. One common model often applies when the machines are arranged in numeric order in a line or semicircle. The robot's travel-time between adjacent machines M_{i-1} and M_i, denoted $d(M_{i-1}, M_i)$, equals δ, for $i = 1, \dots, m+1$, and is additive. Additive means that, for the travel-time between any two machines $M_i, M_j, 0 \leq i, j \leq m+1$, $d(M_i, M_j) = |i - j|\delta$ (this restriction is also known as the *triangle equality*: if $i < k < j$, then $d(M_i, M_j) = d(M_i, M_k) + d(M_k, M_j)$ (Crama et al., 2000)). This scheme is easily generalized to the case of non-equal travel-times between adjacent machines (Brauner and Finke, 1999): $d(M_{i-1}, M_i) = \delta_i, i = 1, \dots, m+1$, and $d(M_i, M_j) = \sum_{k=i+1}^{j} \delta_k$, for $i < j$. If $d(M_{i-1}, M_i) = \delta, i = 1, \dots, m+1$, then we call the travel-time metric *regular additive*. If $d(M_{i-1}, M_i) = \delta_i, i = 1, \dots, m+1$, then the cell has general additive travel-times.

There are also additive travel-time cells in which the machines are arranged in a circle so that I and O are adjacent or in the same location (Drobouchevitch, Sethi, and Sriskandarajah, to appear; Geismar et al., 2004c; Sriskandarajah et al., 2004; Sethi, Sidney, and Sriskandarajah, 2001). In these cells, the robot may travel in either direction to move from one machine to another, e.g., to move from M_1 to M_{m-1}, it may be faster to go via I, O, and M_m, than to go via $M_2, M_3, \dots, M_{m-2}$. For circular cells with regular additive travel-time, $d(M_i, M_j) = \min\{|i-j|\delta, (m+2-|i-j|)\delta\}$. For general additive travel-time cells, $d(M_i, M_j) = \min\{\sum_{k=i+1}^{j} \delta_k, \sum_{k=1}^{i} \delta_k + \delta_{0,m+1} + \sum_{k=j+1}^{m+1} \delta_k\}$, for $i < j$. Throughout the rest of this paper, the additive travel-time metric used will correspond to that used in the study being cited. Most studies assume that travel-times are symmetric $d(M_i, M_j) = d(M_j, M_i), 0 \leq i, j \leq m+1$, and that the travel-time between two machines does not depend on whether or not the robot is carrying a part.

To make this model better represent reality, Logendran and Sriskandarajah (1996) enhance it to account for the robot's acceleration and deceleration. The travel-times between adjacent machines do not change. However, the travel-time between non-adjacent machines is reduced. For each intervening machine, the robot is assumed to save η units of time. Therefore, for $0 \leq i, j \leq m+1$, if $d(M_{i-1}, M_i) = \delta_i$, then

$$d(M_i, M_j) = \sum_{k=\min(i,j)+1}^{\max(i,j)} \delta_k - (|i - j| - 1)\eta.$$

In Sections 6.2 and 10.1, we present formulas with η because this scheme is used by the studies cited.

For certain cells, additive travel-times are not appropriate. Dawande, Sriskandarajah, and Sethi (2002) discuss a type of cell for which the robot travel-time between *any* pair of machines is a constant δ, i.e., $d(M_i, M_j) = \delta, 0 \le i, j \le m+1, i \ne j$. This arises because these cells are compact and their robots move with varying acceleration and deceleration.

The most general model, one that can most accurately represent any robotic cell, assigns a value δ_{ij} for the robot travel-times between any two machines M_i and M_j, $0 \le i, j \le m+1$. These travel-times are, in general, neither additive nor constant. Brauner, Finke, and Kubiak (2003) address this problem by making three rudimentary assumptions that conform to basic properties of Euclidean space:

1. The travel-time from a machine to itself is zero, i.e., $\delta_{ii} = 0, \forall i$.
2. The travel-times satisfy the triangle inequality, i.e., $\delta_{ij} + \delta_{jk} \ge \delta_{ik}, \forall i, j, k$.
3. The travel-times are symmetric, i.e., $\delta_{ij} = \delta_{ji}, \forall i, j$.

A robotic cell that satisfies Assumptions 1 and 2 is called a Euclidean robotic cell; one that satisfies Assumptions 1, 2, and 3 is called a Euclidean symmetric robotic cell. The robot move sequencing problem for either case is NP-hard in the strong sense (Brauner, Finke, and Kubiak, 2003) (see Garey and Johnson (1979) for a description of computational complexity). This result parallels one for the Euclidean traveling salesman problem (Lawler et al., 1985). This is also why most studies approximate reality with additive or constant travel-time models, depending on which of the two fits better.

To summarize, three different robot travel-time metrics have been addressed in the literature: additive, constant, and Euclidean. Each study assumes one of these before deriving its results. Therefore, many results in the field have been proven only for one travel-time metric, rather than for all three.

3.2.3. Number of part-types

If the robotic cell produces identical parts, we refer to it as a *single part-type* cell. In contrast, *multiple part-type* cells process lots that contain different types of parts. Generally, these different types have different processing times for a given machine. Multiple part-type cells are discussed in Section 6. Throughout the rest of the paper, unless otherwise specified, the cell under consideration processes identical parts.

3.3. Objective function

From an optimization point of view, the only objective addressed in the literature is that of maximizing the *throughput*—the long-term average number of completed parts placed into the output buffer per unit time. A precise definition of the throughput is provided in Section 5.

3.4. An $\alpha|\beta|\gamma$ classification for robotic cells

Figure 2 is a pictorial representation of the classification discussed above. A problem is represented using the form $\alpha|\beta|\gamma$, where

(a) $\alpha = RF^g_{m,r}(m_1, \ldots, m_m)$. Here, RF stands for "Robotic Flowshop," m is the number of processing stages, and the vector $(m_1, m_2, \ldots, m_m)$ indicates the number of machines at each stage. When this vector is not specified, $m_i = 1, i = 1, \ldots, m$, and the cell is a simple cell. The second subscript r denotes the number of robots; when not specified, $r = 1$. The superscript g denotes the type of robot used. For example, $g = 1$ (resp. $g = 2$) denotes a single gripper (resp. dual gripper) cell. If g is not specified, then $g = 1$.

(b) $\beta =$ (*pickup, travel-metric, part-type, prod-strategy*) where
- *pickup* $\in$ {*free, no-wait, interval*} specifies the pickup criterion.
- *travel-metric* $\in \{A, C, E\}$ specifies the travel-time metric. "A" (resp. "C" and "E") denotes the additive (resp. constant and Euclidean) travel-time metric.
- If *part-type* is not specified, the cell produces a single part-type; otherwise *part-type* $= MP$ denotes a cell producing multiple part-types.
- *prod-strategy* $\in$ {*cyclic-k*, *all*, CRM} denotes the specific production strategy employed. The detailed descriptions of these strategies appear in later sections so we limit our description here and refer the reader to the corresponding section.
  - (i) In a cell producing either a single part-type or multiple part-types, *cyclic-k* refers to a *cyclic production* strategy wherein exactly k units are produced per cycle (Section 5).
  - (ii) In a cell producing either a single part-type or multiple part-types, *all* refers to a production environment where all production strategies (i.e., cyclic as well as noncyclic) are considered (Sections 5 and 12).
  - (iii) In robotic cells producing multiple part-types (Section 6), CRM refers to the *concatenated robot-move sequence* strategy (Section 6.1).

(c) $\gamma = \mu$ denotes that the objective function to be addressed is that of maximizing the throughput.

We now illustrate our classification with a few examples.

1. $RF_4|(free, A, cyclic\text{-}1)|\mu$ represents a 4-machine simple robotic cell with one single gripper robot, a free-pickup criterion, and additive travel-time metric. It produces a single part-type and operates a cyclic production strategy wherein one unit is produced per cycle. The objective function addressed is maximizing the throughput.
2. $RF_5(1, 4, 2, 3, 2)|(no\text{-}wait, E, cyclic\text{-}2)|\mu$ addresses the problem of maximizing throughput for a 5-stage robotic cell with parallel machines that has 1, 4, 2, 3, and 2 machines, respectively, in stages 1, 2, 3, 4 and 5. The cell produces a single part-type, has one single gripper robot, employs a no-wait pickup criterion and a Euclidean travel-time metric, and produces two units per cycle.
3. $RF^2_{m,3}|(interval, C, \text{MP}, \text{CRM})|\mu$ considers throughput maximization in an m-machine simple robotic cell with three dual gripper robots, an interval pickup criterion, constant travel-time metric, and multiple part-type production using a CRM production strategy.

In the following sections, we use this classification to specify the problem being discussed.

4. CELL DATA

In addition to the robot's travel-time metric, each stage's processing time and the times required for loading and unloading a machine influence the cell's throughput. We now discuss these characteristics and the notation for representing the cell's actions and states. First, we list the basic assumptions throughout most studies:

- All data and events are deterministic.
- All processing is nonpreemptive.
- Parts to be processed are always available at the cell's input device.
- There is always space for completed parts at the output device.
- All data are rational.

4.1. Processing times

Since each of the m stages performs a different function, each, in general, has a different processing time for a given part. For cells with free-pickup or no-wait pickup, the processing time of a machine in stage i is denoted by p_i, $i \in M$. If a cell processes k different types of parts, the processing time of part j on stage i is denoted by p_{ij}, $i \in M$; $j = 1, \ldots, k$. In interval robotic cells, the processing time of machine M_i is specified by a lower bound a_i and an upper bound b_i, e.g., the time that a printed circuit board spends in tank i must be in the interval $[a_i, b_i]$. If multiple part-types are processed in an interval robotic cell, the processing interval for part-type j is denoted by $[a_{ij}, b_{ij}]$.

4.2. Loading and unloading times

Another factor that influences the processing duration for a part is the time required for loading and unloading at each machine. For uniformity, picking a part from I is referred to as unloading $I = M_0$, and dropping a part at O is referred to as loading $O = M_{m+1}$. A common and simple model assumes that the loading and unloading times are equal to ϵ for all machines (Ioachim and Soumis, 1995). More sophisticated models (Brauner and Finke, 2001a) have different values for loads and unloads for each machine: unloading time for M_i is ϵ_i^u, $i = 0, \ldots, m$, and loading time for M_i is ϵ_i^l, $i = 1, \ldots, m + 1$.

4.3. Notations for cell states and robot actions

The concept of an *activity* is widely used in the study of robotic cells. In a simple robotic cell, activity A_i, $i = 0, \ldots, m$, consists of the following sequence:

1. The robot unloads a part from M_i
2. The robot travels from M_i to M_{i+1}
3. The robot loads this part onto M_{i+1}.

Since a part must be processed on all m machines and then placed into the output buffer, one instance of each of the $m + 1$ activities $A_0, A_1, \ldots, A_m$ is required to produce a part.

A notation that represents the state of the cell is also very useful. Information that it must convey includes the status of each machine (occupied or not occupied by a part) and the state of the robot (location and occupancy). This can be presented as an $(m + 1)$-dimensional state space $(e_1, \dots, e_{m+1})$, as shown in Sethi et al. (1992). The first m dimensions each correspond to a machine: $e_i = \phi$ if M_i is unoccupied; $e_i = \Omega$ if M_i is occupied, $i \in M$. The last dimension represents the robot. $e_{m+1} = A_i$ indicates that the robot has just completed activity $A_i, i = 0, \dots, m$.

For $m = 4$, an example state is $(\Omega, \phi, \phi, \Omega, A_3)$: M_2 and M_3 are unoccupied, M_1 and M_4 are occupied, and the robot has just completed loading M_4. From this point, let us now consider what happens if the robot executes activity sequence $A_1 A_2 A_4$: the robot moves to M_1, waits for M_1 to finish processing (if required), unloads a part from M_1, travels to M_2, and loads the part onto M_2. At this instant, the state of the cell is $(\phi, \Omega, \phi, \Omega, A_1)$. The robot waits at M_2 for the entirety of the part's processing. The robot then unloads the part from M_2, carries it to M_3, and loads the part onto M_3. The cell's state is now $(\phi, \phi, \Omega, \Omega, A_2)$. The robot next travels to M_4, waits for M_4 to finish processing (if required), unloads a part from M_4, travels to the output buffer, and loads the part onto the output buffer, so the cell's state is $(\phi, \phi, \Omega, \phi, A_4)$.

It is important to note that this state description is not a complete definition of the state, as it omits information representing the extent of the processing completed on the parts on the various machines. However, as we are mainly concerned about cyclic solutions (defined in Section 5), we see later in Section 5.1.2 that a condition that defines cyclic solutions completes the missing information in the state description. Thus, for our purposes, this state description is sufficient.

5. CYCLIC PRODUCTION

Cyclic production in a robotic cell refers to the production of finished parts by repeating a fixed sequence of robot moves. The main motivation for studying cyclic production comes from practice: cyclic schedules are easy to implement and control and are the primary way of specifying the operation of a robotic cell in industry. In a recent result, Dawande, Geismar, and Sethi (to appear) show that it is sufficient to consider cyclic schedules in order to maximize throughput. That is, there is at least one cyclic schedule in the set of all schedules that optimizes the throughput of the cell. The main idea behind this result is easy to explain: a robotic cell can be modeled as a finite state dynamic system. Any potential throughput-maximizing way of operating the cell can be represented by a policy (function) defined on the finite state space. The finiteness of the state space then implies that any particular policy repeats a minimal sequence of robot moves and, thus, gives rise to a cyclic schedule.

Using the notation defined in the previous section, cyclic production can be represented as a repeatable sequence of activities. For example, $(A_0, A_2, A_4, A_3, A_1)$ is a sequence of activities that produces a part in a four-machine cell. Such a sequence can be repeated in a cyclic fashion, with each iteration producing a single part. To formalize, we define the following terms:

Definition. A *k-unit activity sequence* is a sequence of robot moves which loads and unloads each machine exactly k times.

To be feasible, an activity sequence must satisfy two criteria:

- The robot cannot be instructed to load an occupied machine
- The robot cannot be instructed to unload an unoccupied machine.

These concepts are operationalized as follows: *During cyclic operations, for* $i = 1, \dots, m - 1$, *between any two occurrences of* A_i *there must be exactly one* A_{i-1} *and exactly one* A_{i+1}. This condition implies that between any two instances of A_0 there is exactly one A_1, and between any two instances of A_m there is exactly one A_{m-1}. For instance, in a cell with $m = 3$, the 2-unit activity sequence (A_0, A_1, A_3, A_1, A_2, A_0, A_3, A_2) is infeasible because the second occurrence of A_1 attempts to unload machine M_1 when it is empty. Note that all 1-unit activity sequences are feasible.

Definition. A *k-unit cycle* is the performance of a feasible k-unit activity sequence in a way which leaves the cell in exactly the same state as its state at the beginning of those moves.

For every feasible k-unit activity sequence, $k \geq 1$, there is at least one initial state for which it is a k-unit cycle, i.e., if the k-unit activity sequence begins with this state, it leaves the cell in exactly the same state after its execution (Sriskandarajah et al., 2004). Since a k-unit cycle preserves the state of the cell, repeating it indefinitely yields a *k-unit cyclic solution*. A cyclic solution is also known as a *steady state* solution. We provide a more rigorous definition of *steady state* below.

A k-unit activity sequence has $k(m+1)$ activities, i.e., each of the $m+1$ activities is performed exactly k times and in an order that satisfies the feasibility constraints. A k-unit cycle constructed from a k-unit activity sequence (A_0, A_{i_1}, A_{i_2}, ..., $A_{i_{k(m+1)-1}}$) will be referred to as the k-unit cycle (A_0, A_{i_1}, A_{i_2}, ..., $A_{i_{k(m+1)-1}}$) or, simply, cycle (A_0, A_{i_1}, A_{i_2}, ..., $A_{i_{k(m+1)-1}}$) . Since a k-unit cyclic solution is completely characterized by a k-unit cycle, we will use the two terms interchangeably when no confusion arises by doing so.

Define the function $S(A_i, t)$ to represent the time of completion of the t^{th} execution of activity A_i (Crama and van de Klundert, 1997a). Given a feasible infinite sequence of activities and a compatible initial state, we can define the *long-run average throughput* or, simply, *throughput* to be

$$\mu = \lim_{t\to\infty} \frac{t}{S(A_m, t)}.$$

Intuitively, this quantity represents the long-term average number of completed parts placed into the output buffer per unit time.

Obtaining a feasible infinite sequence of activities that maximizes throughput is a fundamental problem of robotic cell scheduling. Such a sequence of robotic moves is called *optimal*. Most studies focus on infinite sequences of activities in which a fixed sequence of $m+1$, or some integral multiple of $m+1$, activities is repeated cyclically.

Definition (Crama and van de Klundert, 1997a). A robotic cell repeatedly executing a k-unit cycle π of robot moves is operating in *steady state* if there exists a constant $T(\pi)$ and a constant N such that for every A_i, $i = 0, \ldots, m$, and for every $t \in \mathbb{Z}^+$ such that $t > N$, $S(A_i, t+k) - S(A_i, t) = T(\pi)$. $T(\pi)$ is called the *cycle time* of π.

For additive travel-time cells, we denote the cycle time by $T_a(\pi)$. For constant travel-time cells, we denote the cycle time by $T_c(\pi)$. For Euclidean travel-time cells, we denote the cycle time by $T_e(\pi)$.

The *per unit cycle time* of a k-unit cycle π is $\frac{T(\pi)}{k}$. This is the reciprocal of the throughput and is easier to calculate directly. Therefore, minimizing the per unit cycle time is equivalent to maximizing the throughput.

An assumption in most studies is that the sequence of robot moves is *active*.

Definition. A sequence is called *active* if the robot always executes the next operation, whatever that may be, as soon as possible.

For active sequences, all execution times for the robot's actions are uniquely determined once the sequence of activities is given. The robot's only possible waiting period can occur at a machine at which the robot has arrived to unload, but the machine has not completed processing its current part. It is known that there always exists an active sequence that is optimal within the class of 1-unit cycles (van de Klundert, 1996).

Brauner and Finke (2001b) show that repeating a k-unit activity sequence will enable the robotic cell to reach a steady state (or cyclic solution) in finite time. Therefore, since we are maximizing the long-run average throughput, i.e., assuming that the cell operates in steady state for an infinite time, there is no impact from the initial transient phase (Hall, Kamoun, and Sriskandarajah, 1998; Dawande, Geismar, and Sethi, to appear). Hence, there is no loss of generality by studying only the steady state behavior. Nevertheless, there may be some practical reason to find the time required to reach steady state. This is discussed in Section 10.1.

5.1. *Cycle times*

In this section we discuss the robot's waiting time at a particular machine and a method of finding the cycle time of a 1-unit cycle in a simple robotic cell. We also establish lower bounds for the cycle time.

5.1.1. *Waiting times*

The robot waits at a machine M_i if its next sequenced action is to unload M_i, but M_i has not yet completed processing its current part. The length of the robot's *waiting time*, denoted w_i, at machine M_i, $i \in M$, is M_i's processing time p_i minus the time that elapses between when M_i was loaded and when the robot returns to unload it. If this difference is negative, then the waiting time is zero.

The time that elapses between M_i's loading and the robot's return is determined by the intervening activities that are executed between the loading and the unloading of M_i. If there are no intervening activities, the robot loads M_i, waits at M_i for time p_i, then unloads M_i. Such a sequence is represented by $A_{i-1}A_i$. In this case, M_i is said to have *full waiting* (Dawande, Sriskandarajah, and Sethi, 2002).

If there are intervening activities between the loading and the unloading of M_i, then M_i has *partial waiting* (Dawande, Sriskandarajah, and Sethi, 2002). Consider the sequence $A_{i-1}A_jA_i$. The robot loads M_i, travels to M_j ($\delta_{i,j}$), waits for M_j to complete processing (w_j), unloads M_j (ϵ_j^u), carries that part to M_{j+1} ($\delta_{j,j+1}$), loads M_{j+1} (ϵ_{j+1}^l), then travels to M_i ($\delta_{j+1,i}$). The robot's waiting time at M_i is

$$w_i = \max\left\{0,\, p_i - \delta_{i,j} - w_j - \epsilon_j^u - \delta_{j,j+1} - \epsilon_{j+1}^l - \delta_{j+1,i}\right\}.$$

For a constant travel-time cell, this expression simplifies to $w_i = \max\{0, p_i - 3\delta - 2\epsilon - w_j\}$.

The expression for the robot's waiting time is often dependent on the waiting time at another machine. This recursion makes calculating the cycle time more difficult. However, the condition that a cycle begins and ends in the same state allows us to uniquely compute the cycle times, as demonstrated in the next section.

5.1.2. *Computing cycle times*

The cycle time is calculated by summing the robot's movement times, the loading and unloading times, and the robot's waiting times (full and partial). Unlike the linear programming approach used in Crama et al. (2000), we need not directly deduce the exact time at which each machine is loaded and unloaded.

For each activity $A_i, i = 0, \ldots, m$, the robot unloads M_i, carries the part to M_{i+1}, and loads M_{i+1}. The total time for A_i is $\epsilon_i^u + \delta_{i,i+1} + \epsilon_{i+1}^l$.

We must also account for the time between activities. If M_i has full waiting (A_{i-1} immediately precedes A_i), the robot spends exactly p_i time units between activities A_{i-1} and A_i waiting at M_i. If M_i has partial waiting (A_j immediately precedes A_i, $j \neq i-1$), then the robot moves from M_{j+1} to M_i ($\delta_{j+1,i}$) and waits for M_i to complete processing (w_i) before starting activity A_i.

For clarity, we now assume a constant travel-time cell with constant loading and unloading times. Let V_1 be the set of machines with full waiting, and V_2 be the set of those with partial waiting. The cycle time for a 1-unit cycle is

$$T_c(\pi) = (m+1)(\delta + 2\epsilon) + \sum_{i \in V_1} p_i + \sum_{i \in V_2} (w_i + \delta) + \delta. \tag{1}$$

The extra δ accounts for the last movement of the cycle, which takes the robot to I to collect a new part.

For example, consider the cycle $S_3 = (A_0, A_1, A_3, A_2)$. $V_1 = \{1\}$, $V_2 = \{2, 3\}$, $m = 3$.

$$\begin{aligned}
T_c(S_3) &= 4(\delta + 2\epsilon) + p_1 + w_2 + w_3 + 3\delta \\
&= 7\delta + 8\epsilon + p_1 + w_2 + w_3 \\
w_2 &= \max\{0, p_2 - 3\delta - 2\epsilon - w_3\} \\
w_3 &= \max\{0, p_3 - 4\delta - 4\epsilon - p_1\} \\
w_2 + w_3 &= \max\{0, p_2 - 3\delta - 2\epsilon, p_3 - 4\delta - 4\epsilon - p_1\} \\
T_c(S_3) &= \max\{7\delta + 8\epsilon + p_1, 4\delta + 6\epsilon + p_1 + p_2, 3\delta + 4\epsilon + p_3\}
\end{aligned}$$

Similarly, the cycle time for the cycle $S_6 = (A_0, A_3, A_2, A_1)$ is $T_c(S_6) = \max\{8\delta + 8\epsilon, p_1 + 3\delta + 4\epsilon, p_2 + 3\delta + 4\epsilon, p_3 + 3\delta + 4\epsilon\}$. Writing the equations for the waiting times requires that the cycle begins and ends in the same state. In general, this method can be implemented as a linear program with km variables and km constraints, where k is the number of units produced in one cycle (Geismar, Dawande, and Sriskandarajah, 2004b; Kumar, Ramanan, and Sriskandarajah, 2005). Hence, it has complexity $O((km)^3 L)$, where L is the size of the problem's binary encoding.

Crama and van de Klundert (1997a) develop an $O(m)$ time algorithm to find the cycle time in an additive travel-time cell of any member of a dominant subset of cycles called *pyramidal cycles*. Pyramidal cycles are discussed in greater detail in Section 5.2.

There are alternative graphical methods that find cycle times without considering robot waiting times. We refer the reader to Crama et al. (2000) for an extensive description of these.

5.1.3. Lower bounds on cycle times

From equation (1) we can deduce a lower bound for the cycle time for a 1-unit cycle in a constant travel-time cell (problem $RF_m|(free, C, cyclic\text{-}1)|\mu$). Obviously, for any cycle, $T_c(\pi) \geq 2(m+1)\epsilon + (m+2)\delta$. If all machines with partial waiting have $w_i = 0$, then the minimum value for $T_c(\pi)$ is achieved by minimizing $\sum_{i \in V_1} p_i + \sum_{i \in V_2} \delta$, which is done by placing those machines for which $p_i \leq \delta$ in V_1. Thus, in a constant travel-time robotic cell, for any 1-unit cycle π, $T_c(\pi) \geq (m+2)\delta + \sum_{i=1}^{m} \min\{p_i, \delta\} + 2(m+1)\epsilon$ (Dawande, Sriskandarajah, and Sethi, 2002). In a regular additive travel-time robotic cell (problem $RF_m|(free, A, cyclic\text{-}1)|\mu$), for any 1-unit cycle π, $T_a(\pi) \geq 2(m+1)(\delta+\epsilon)+\sum_{i=1}^{m}\min\{p_i,\delta\}$ (Crama and van de Klundert, 1997a).

Suppose that $p_j = \max_{1\leq i\leq m} p_i$ is large relative to δ and ϵ. Since the cycle time can be measured as the time between successive loadings of M_j, we can derive another lower bound for the cycle time of a 1-unit cycle. This includes, at minimum, the times for the following: M_j's processing, unload M_j, move to M_{j+1}, load M_{j+1}, move to M_{j-1}, unload M_{j-1}, move to M_j, load M_j. For constant travel-time, this value is $p_j + 3\delta + 4\epsilon$; for regular additive travel-time, $p_j + 4(\delta+\epsilon)$. We combine these bounds, originally derived by Dawande, Sriskandarajah, and Sethi (2002) and Crama and van de Klundert (1997a), respectively, in the following theorem.

Theorem 1. For 1*-unit cycles, the following are lower bounds for constant travel-time robotic cells (problem* $RF_m|(free, C, cyclic\text{-}1)|\mu$) *and regular additive travel-time robotic cells (problem* $RF_m|(free, A, cyclic\text{-}1)|\mu$), *respectively*:

$$T_c(\pi) \geq \max\left\{(m+2)\delta + \sum_{i=1}^{m}\min\{p_i,\delta\} + 2(m+1)\epsilon,\ \max_{1\leq i\leq m} p_i + 3\delta + 4\epsilon\right\}$$

$$T_a(\pi) \geq \max\left\{2(m+1)(\delta+\epsilon) + \sum_{i=1}^{m}\min\{p_i,\delta\},\ \max_{1\leq i\leq m} p_i + 4(\delta+\epsilon)\right\}.$$

5.2. Optimal 1-unit cycles

We first examine two elementary cycles on simple robotic cells with free-pickup and then examine specific conditions under which they are optimal. We then discuss two classes of cycles in which an optimal cycle can be found under more general conditions for cells with free-pickup, and summarize an approach to finding optimal cycles for no-wait cells.

5.2.1. Special cases

In the forward cycle $\pi_U = (A_0, A_1, A_2, \ldots, A_{m-1}, A_m)$, often called S_1, the robot unloads a part from I, carries it to M_1, loads M_1, waits for M_1 to process the part, unloads M_1, then carries the part to M_2. The robot continues in this fashion, waiting at each machine for its entire processing of the part. Only one machine is processing a part at any given time. The processing times for π_U in constant and regular additive travel-time robotic cells, respectively, are

$$T_c(\pi_U) = 2(m+1)\epsilon + \sum_{i=1}^{m} p_i + (m+2)\delta$$

$$T_a(\pi_U) = 2(m+1)\epsilon + \sum_{i=1}^{m} p_i + 2(m+1)\delta.$$

For constant and additive travel-time simple robotic cells, Theorems 2–4 below provide an optimal 1-unit cycle under specific conditions. In terms of the classification provided in Section 3.4, these results are for problems $RF_m|(free, C, cyclic\text{-}1)|\mu$ and $RF_m|(free, A, cyclic\text{-}1)|\mu$.

Theorem 2. For both constant and regular additive travel-time robotic cells, if $p_i \leq \delta, \forall i$, *then* π_U *achieves the optimal* 1*-unit cycle time.*

This result follows immediately from Theorem 1 (Dawande, Sriskandarajah, and Sethi, 2002).

The reverse cycle for a simple robotic cell, often called $S_{m!}$, is $\pi_D = (A_0, A_m, A_{m-1}, \dots, A_2, A_1)$. To perform π_D, the robot unloads a part from the input buffer (M_0), carries it to M_1, and loads M_1. It then travels to M_m, unloads M_m, and carries that part to the output buffer (M_{m+1}). It repeats this sequence for $i = m-1, m-2, \dots, 1$: travel to M_i, unload M_i, carry the part to M_{i+1}, load M_{i+1}. After loading M_2 (which completes activity A_1), the robot completes the cycle by traveling to the input buffer (M_0). At each machine, before unloading a part from it, the robot may have to wait for that machine to complete processing.

The cycle times for π_D in constant (Dawande, Sriskandarajah, and Sethi, 2002) and regular additive (Crama and van de Klundert, 1997a) travel-time robotic cells, respectively, are

$$T_c(\pi_D) = \max\left\{2(m+1)(\delta+\epsilon),\ \max_{1\le i\le m} p_i + 3\delta + 4\epsilon\right\}$$

$$T_a(\pi_D) = \max\left\{4m\delta + 2(m+1)\epsilon,\ \max_{1\le i\le m} p_i + 4(\delta+\epsilon)\right\}.$$

Note that in each formula, the first argument represents the cycle time if the robot never waits for a machine to complete its processing.

For each of the following two theorems, if its premises are met, then π_D achieves the lower bound stated in Theorem 1. This yields the following theorem from Dawande, Sriskandarajah, and Sethi (2002) and Crama and van de Klundert (1997a):

Theorem 3. In a constant travel-time robotic cell (*problem* $RF_m|(free, C, cyclic\text{-}1)|\mu$), *if* $\max_{1\le i\le m} p_i + 3\delta + 4\epsilon \ge 2(m+1)(\delta+\epsilon)$, *then* π_D *is an optimal* 1-*unit cycle. In a regular additive travel-time robotic cell* (*problem* $RF_m|(free, A, cyclic\text{-}1)|\mu$), *if* $\max_{1\le i\le m} p_i + 4(\delta+\epsilon) \ge 4m\delta + 2(m+1)\epsilon$, *then* π_D *is an optimal* 1-*unit cycle.*

Theorem 3 can be generalized to the Euclidean travel-time case (problem $RF_m|(free, E, cyclic\text{-}1)|\mu$). If

$$\max_{1\le i\le m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\} \ge 2(m+1)\epsilon + \sum_{i=0}^{m}\delta_{i,i+1} + \sum_{i=2}^{m+1}\delta_{i,i-2} + \delta_{1,m}, \tag{2}$$

then π_D is optimal. If condition (2) holds, then $T_e(\pi_D) = \max_{1\le i\le m}\{p_i + \delta_{i,i+1} + \delta_{i+1,i-1} + \delta_{i-1,i} + 4\epsilon\}$, which, by following the logic of Section 5.1.3 and using the triangle inequality, we see is a lower bound on the cycle time.

The following theorem is proven by Dawande, Sriskandarajah, and Sethi (2002).

Theorem 4. For constant travel-time robotic cells (*problem* $RF_m|(free, C, cyclic\text{-}1)|\mu$), *if* $p_i \ge \delta, \forall i$, *then* π_D *achieves the optimal* 1-*unit cycle time.*

This theorem is not true for additive travel-time robotic cells. Consider the following example: $\pi_1 = (A_0, A_1, A_m, A_{m-1}, \dots, A_2)$, with $p_i \ge \delta, \forall i$.

$$T_a(\pi_1) = \max\{(4m-2)\delta + 2(m+1)\epsilon + p_1,\ p_2 + p_1 + 6(\delta+\epsilon),\ \max_{3\le i\le m}\{p_i + 4(\delta+\epsilon)\}\} \tag{3}$$

If

$$p_2 + p_1 + 6(\delta + \epsilon) \leq (4m-2)\delta + 2(m+1)\epsilon + p_1, \quad \text{and}$$
$$\max_{3\leq i\leq m}\{p_i + 4(\delta+\epsilon)\} \leq (4m-2)\delta + 2(m+1)\epsilon + p_1,$$

then $T_a(\pi_1) = (4m-2)\delta + 2(m+1)\epsilon + p_1$. If $p_1 < 2\delta$, then $T_a(\pi_D) = 4m\delta + 2(m+1)\epsilon$, and $T_a(\pi_1) < T_a(\pi_D)$. However, we do have the following results for regular additive travel-time cells.

Theorem 5. For problem $RF_m|(free, A, cyclic\text{-}1)|\mu$, if

$$p_i + p_{i+1} \geq (4m-6)\delta + 2(m-2)\epsilon, \quad i = 1, \ldots, m-1,$$

then π_D is optimal.

Proof. See the Appendix. ■

Corollary 1. For problem $RF_m|(free, A, cyclic\text{-}1)|\mu$, if

$$p_i \geq (2m-3)\delta + (m-2)\epsilon, \quad i = 1, \ldots, m,$$

then π_D is optimal.

5.2.2. General cases

To find an optimal 1-unit cycle in additive travel-time cells (problem $RF_m|(free, A, cyclic\text{-}1)|\mu$), Crama and van de Klundert (1997a) employ a concept that has been used to analyze traveling salesman problems: the set of 1-unit pyramidal cycles (Lawler et al., 1985).

Definition. The 1-unit cycle $\pi = (A_0, A_{i_1}, A_{i_2}, \ldots, A_{i_m})$ is pyramidal if there exists a $k \in M$ such that $1 \leq i_1 < i_2 < \cdots < i_k = m$, and $m > i_{k+1} > i_{k+2} > \cdots > i_m \geq 1$. In such a cycle, $U = \{i_1, i_2, \ldots, i_k\}$ is the set of uphill activities and $D = \{i_{k+1}, i_{k+2}, \ldots, i_m\}$ is the set of *downhill activities.*

π_U and π_D are pyramidal, as is $(A_0, A_2, A_5, A_7, A_6, A_4, A_3, A_1)$. In an m-machine cell, there are 2^{m-1} pyramidal cycles. Crama and van de Klundert (1997a) show that the set of pyramidal cycles is dominant over all 1-unit cycles in additive travel-time cells.

Definition. A set of cycles Π is *dominant* if, for every choice of the processing times, there exists $\pi \in \Pi$ such that $T(\pi)/k_\pi \leq T(\pi')/k_{\pi'}, \forall \pi' \notin \Pi$, where π is a k_π-unit cycle, and π' is a $k_{\pi'}$-unit cycle.

They also devise a dynamic programming algorithm that finds the optimal pyramidal permutation in $O(m^3)$ time (Crama and van de Klundert, 1997a).

Pyramidal cycles do not dominate the class of 1-unit cycles in constant travel-time cells for $m \geq 3$ (problem $RF_m|(free, C, cyclic\text{-}1)|\mu$), but the set of *basic* cycles does. To create a *basic* cycle for a cell, one first must decide which machines M_i have full waiting ($i \in V_1$), and which have partial waiting ($i \in V_2$). Each element of the set $\{A_i : i \in V_2 \cup \{0\}\}$ is the first element of a string $\tau_i = \{A_i, A_{i+1}, \ldots, A_{i+l_i}\}$. All other elements of a string correspond to machines with full waiting: both $i + k \in V_1, k = 1, \ldots, l_i$, and $i + l_i + 1 \in V_2 \cup \{m+1\}$, for $i \in V_2 \cup \{0\}$. The basic cycle is formed by concatenating the strings in reverse order (Dawande, Sriskandarajah, and Sethi, 2002). Consider the following example:

Example. $m = 8$, $V_1 = \{1, 2, 4, 8\}$, and $V_2 = \{3, 5, 6, 7\}$. There are five strings: $\tau_0 = \{A_0, A_1, A_2\}$, $\tau_3 = \{A_3, A_4\}$, $\tau_5 = \{A_5\}$, $\tau_6 = \{A_6\}$, and $\tau_7 = \{A_7, A_8\}$. The basic 1-unit cycle corresponding to V_1 is $(\tau_0, \tau_7, \tau_6, \tau_5, \tau_3) = (A_0, A_1, A_2, A_7, A_8, A_6, A_5, A_3, A_4)$.

In a cell with m machines, there are $2^m - m$ basic cycles. To find the optimal basic cycle, Dawande, Sriskandarajah, and Sethi (2002) devise an $O(m^2 \log(m\delta))$ algorithm based on repeatedly solving a shortest path problem in a directed acyclic network.

For no-wait cells (Levner, Kats, and Levit, 1997), develop a polynomial algorithm for finding the minimum cycle time for Euclidean travel-time cells (problem $RF_m|(no\text{-}wait,\ E,\ cyclic\text{-}1)|\mu$). It uses the machine's processing times and the robot's travel-times to derive infeasible intervals for the cycle time. The optimal cycle time is the smallest positive number not in these intervals. Obviously, this algorithm can be applied to the less general cases of additive (problem $RF_m|(no\text{-}wait,\ A,\ cyclic\text{-}1)|\mu$) and constant (problem $RF_m|(no\text{-}wait,\ C,\ cyclic\text{-}1)|\mu$) travel-times, too.

5.3. Quality of 1-unit cycles and approximation results

Having found optimal cycles for problems $RF_m|(free,\ A, cyclic\text{-}1)|\mu$ and $RF_m|(free,\ C,\ cyclic\text{-}1)|\mu$, the question naturally arises, "Is the optimal 1-unit cycle superior to every nontrivial k-unit cycle, $k \geq 2$?" Sethi et al. (1992) prove this to be true for $RF_2|(free,\ A,\ cyclic\text{-}1)|\mu$ and conjectured it to be so for $m \geq 3$. The attraction of this possibility is obvious: 1-unit cycles are the easiest to understand, analyze, and control. If they also have the highest throughput, there is no reason to consider the more complex and more numerous multiunit cycles.

For $RF_3|(free,\ A, cyclic\text{-}1)|\mu$, Crama and van de Klundert (1999) and Brauner and Finke (1999a) each prove that the Sethi et al. conjecture is true. The conjecture does not hold for $RF_4|(free, A, cyclic\text{-}1)|\mu$, however. Brauner and Finke (1997, 2001b) provide a counterexample. For $RF_4|(free, C, cyclic\text{-}1)|\mu$, consider the cell with the following data (Dawande, Geismar, and Sethi, to appear): $p_1 = 22$, $p_2 = 1$, $p_3 = 1$, $p_4 = 22$, $\delta = 4$, $\epsilon = 0$. The best 1-unit cycle is $(A_0, A_4, A_3, A_1, A_2)$, whose cycle time is 39. The best 2-unit cycle time is achieved by $(A_0, A_4, A_3, A_1, A_0, A_4, A_2, A_3, A_1, A_2)$, whose per unit cycle time is 38. Note that although this 2-unit cycle dominates all 1-unit cycles and all 2-unit cycles in this cell, we cannot assert its optimality for all k-unit cycles, $k \geq 1$. By Theorem 1, the lower bound on the optimal value is 34, so there may be a k-unit cycle, $k \geq 3$, that has per unit cycle time less than 38.

Similar results for $RF_m|(no\text{-}wait,\ A,\ cyclic\text{-}1)|\mu$ and $RF_m|(interval,\ A,\ cyclic\text{-}1)|\mu$ are summarized in Crama et al. (2000).

Even though 1-unit cycles do not dominate, their simplicity still makes them attractive in practice. We have seen that the reverse cycle π_D is optimal under certain conditions. Crama and van de Klundert (1997a) show that for $RF_m|(free,\ A,\ cyclic\text{-}1)|\mu$, π_D is a 2-approximation. Brauner and Finke (2001a) show that if the optimum per unit cycle time over all k-unit cycles is T_{opt}, then one can guarantee that

$$T_{\text{a}}(\pi_D) \leq \left(2 - \frac{\delta_1 + \delta_{m+1}}{\delta_1 + \delta_{m+1} + \sum_{i=2}^{m} \delta_i}\right) T_{\text{opt}} \leq 2T_{\text{opt}}.$$

For $RF_m|(free, C, cyclic\text{-}1|\mu$, we have

$$T_c(\pi_D) \leq \left(\frac{2(m+1)(\delta+\epsilon)}{(m+2)\delta+2(m+1)\epsilon}\right) T_{\text{opt}} \leq \left(\frac{2(m+1)\delta}{(m+2)\delta}\right) T_{\text{opt}} \leq 2T_{\text{opt}}.$$

The most recent approximation algorithms for 1-unit cycles in additive, constant, and Euclidean travel-time cells are by Geismar, Dawande, and Sriskandarajah (2005). For both problem $RF_m|(free, A, cyclic\text{-}1)|\mu$ and problem $RF_m|(free, C, cyclic\text{-}1)|\mu$, they develop algorithms that produce 1-unit cycles whose cycle times are within a factor of 1.5 of the optimum per unit cycle time. For Euclidean travel-time cells, where the optimum 1-unit cycle problem (i.e. problem $RF_m|(free, E, cyclic\text{-}1)|\mu$) is NP-complete (Brauner, Finke, and Kubiak, 2003), they develop an algorithm that produces a 1-unit cycle whose cycle time is within a factor of $1.5q$ of the optimum per unit cycle time, where

$$q = \frac{\max_{0\leq i,j\leq m+1}\{\delta_{ij}\}}{\min_{0\leq i,j\leq m+1}\{\delta_{ij}\}}, \quad i \neq j.$$

Should this ratio be large ($q \geq 2.67$), it is also shown that π_D is a 4-approximation. All three algorithms run in $O(m)$ time.

An approximation result for dual gripper robot cells (problem $RF_m^2|(free, A, \text{MP}, \text{CRM})|\mu$) can be found in Section 7. Heuristics for $RF_m|(free, A, \text{MP}, cyclic\text{-}k)|\mu$ are described in Section 6.3.

6. MULTIPLE PART-TYPES

We now examine robotic cells that process lots that contain different types of parts. Generally, parts of different types have different processing times for a given machine. Such implementations are more common in smaller manufacturers. They process multiple parts in a single lot in order to have enough volume to use the cell efficiently (Ramanan, 2002).

In accordance with just-in-time manufacturing, the relative proportions of the part-types in each lot should be the same as the relative proportions of the demand. Consequently, researchers focus on cycles which contain the minimal part set (MPS) that has these same proportions. For example, if the demand for a company's three products is divided so that product A has 40%, product B has 35%, and product C has 25%, the MPS has 20 parts: 8 of product A, 7 of product B, and 5 of product C. In practice, the size of an MPS can be larger than 50 parts (Wittrock, 1985).

In general, the cell processes k different part-types. In one MPS, r_i parts of type i are produced, $i = 1, \ldots, k$. The total number of finished parts in a cycle is $n = r_1 + \cdots + r_k$. The schedule according to which the parts are produced is specified by a permutation σ. $P_{\sigma(i)}$ is the part scheduled in the i^{th} position of σ, $i = 1, \ldots, n$.

6.1. *MPS cycles and CRM sequences*

Given an MPS of n parts, an MPS *cycle* is a sequence of robot moves in which exactly n parts of an MPS enter the cell at I, exactly n parts of an MPS exit the cell at O, and the cell returns to its initial state. The order in which the parts enter the cell is called the MPS *part schedule* (or simply *part schedule*). An MPS cycle is determined by the MPS part schedule and the MPS *robot move sequence* (or simply *robot move sequence*) that specifies all robot operations during the MPS cycle. Observe that single part-type production is a special case in which $n = 1$.

Concatenated Robot Move Sequences (CRM sequences) form a class of MPS cycles in which the robot move sequence is the same 1-unit cycle of robot actions repeated n times (Sriskandarajah et al., 2004). For example, for $m = 3$, the CRM sequence based on $S_4 = (A_0, A_3, A_1, A_2)$ for $n = 3$ is $(S_4, S_4, S_4) = (A_0, A_3, A_1, A_2, A_0, A_3, A_1, A_2, A_0, A_3, A_1, A_2)$.

6.2. Elementary results (m = 2)

Optimizing a cell that produces multiple part-types requires that two intertwined problems be solved: find the optimal MPS part schedule, and find the optimal MPS robot move sequence. Most studies fix the robot move sequence to be a specific CRM sequence and then find the MPS part schedule that minimizes the total cycle time (problem $RF_m|(free, A, \text{MP}, \text{CRM})|\mu$). The best part schedules for each CRM sequence are then compared, where $\mu = n/\mathit{cycle\ time}$.

For $m = 2$, in the CRM sequence corresponding to forward cycle $\pi_U = S_1 = (A_0, A_1, A_2)$, the cycle time is independent of the part schedule (Sethi et al., 1992). Logendran and Sriskandarajah (1996) show that the cycle time for a robotic cell with general additive travel-time is

$$T_a(\pi_U) = n\left(\sum_{i=0}^{2}\epsilon_i^u + \sum_{i=1}^{3}\epsilon_i^l + 2\sum_{i=1}^{3}\delta_i - 2\eta\right) + \sum_{j=1}^{n}(p_{1j} + p_{2j}).$$

For constant travel-time robotic cells, we have

$$T_c(\pi_U) = n(4\delta + 6\epsilon) + \sum_{j=1}^{n}(p_{1j} + p_{2j}).$$

For the CRM sequence corresponding to reverse cycle $\pi_D = S_2 = (A_0, A_2, A_1)$ in a two-machine cell, the optimal part schedule can be found by formulating the problem as a special case of a traveling salesman problem which can be solved by using the Gilmore and Gomory (1964) algorithm for no-wait flowshops (Logendran and Sriskandarajah, 1996):

$$T_a(\pi_D) = n\left(2\delta_2 + \epsilon_1^u + \epsilon_2^l\right) + \max\left\{e_{\sigma(2)}, f_{\sigma(1)}\right\} + \max\left\{e_{\sigma(3)}, f_{\sigma(2)}\right\} + \cdots + \max\left\{e_{\sigma(1)}, f_{\sigma(n)}\right\},$$

where

$$e_j = p_{1j} + 2\delta_1 + \epsilon_0^u + \epsilon_1^l - \eta, \quad \text{and}$$

$$f_j = \max\left\{2\left(\sum_{i=1}^{3}\delta_i - \eta\right) + \epsilon_0^u + \epsilon_1^l + \epsilon_2^u + \epsilon_3^l,\ p_{2j} + 2\delta_3 + \epsilon_2^u + \epsilon_3^l - \eta\right\}.$$

For constant travel-time robotic cells, we have

$$T_c(\pi_D) = 4n(\delta + \epsilon) + \max\left\{e_{\sigma(2)}, f_{\sigma(1)}\right\} + \max\left\{e_{\sigma(3)}, f_{\sigma(2)}\right\} + \cdots + \max\left\{e_{\sigma(1)}, f_{\sigma(n)}\right\},$$

where $e_j = p_{1j}$ and $f_j = \max\{4\delta + 2\epsilon, p_{2j}\}$.

For additive travel-time cells (problem $RF_2|(free, A, \text{MP}, cyclic\text{-}k)|\mu$), (Hall, Kamoun, and Sriskandarajah, 1997) attack the two problems—part scheduling and robot move sequencing—simultaneously. They first show that, in general, CRM sequences are not optimal MPS robot move sequences. Rather, it is often better to selectively switch between S_1 and S_2.

In both cycles S_1 and S_2, a pair of parts $P_{\sigma(i)}$ and $P_{\sigma(i+1)}$ is involved. The state (ϕ, Ω, A_1), in which the robot has just finished loading a part onto M_2, is the only state common to both S_1 and

S_2. Hence, this is the only state in which switching between S_1 and S_2 can be achieved without wasteful robot moves. The robot has two choices for its next action from this state:

(i) wait and unload M_2 (as in cycle S_1), or
(ii) move to I (as in cycle S_2).

Because of the possibility of switching between cycles at state (ϕ, Ω, A_1), the robot may be performing one cycle while part $P_{\sigma(i)}$ is being processed on M_1 and the other cycle while $P_{\sigma(i)}$ is being processed on M_2. If the robot uses S_1 (resp. S_2) while $P_{\sigma(i)}$ is processed on M_1, and S_2 (resp. S_1) while $P_{\sigma(i)}$ is processed on M_2, then we say that $P_{\sigma(i)}$ is processed using cycle $S_{1,2}$ (resp. $S_{2,1}$). The cycle that is used while $P_{\sigma(i)}$ is on M_2 must also be used while $P_{\sigma(i+1)}$ is on M_1. Hence, if $P_{\sigma(i)}$ is processed using S_1 or $S_{2,1}$ (resp. S_2 or $S_{1,2}$), then $P_{\sigma(i+1)}$ must be processed using S_1 or $S_{1,2}$ (resp. S_2 or $S_{2,1}$). For example, an MPS of five parts could be produced by the following cycle of robot move cycles: $S_{1,2}$, S_2, S_2, $S_{2,1}$, S_1. Hall, Kamoun, and Sriskandarajah (1997) find this optimal cycle using an $O(n^4)$ algorithm. Aneja and Kamoun (1999) improve upon this by providing an $O(n \log n)$ algorithm.

6.3. Complexity results ($m \geq 3$)

The complexity results of this section concern CRM sequences. All were developed for additive travel-time cells (problem $RF_m|(free, A, \text{MP}, \text{CRM})|\mu$), but these results are also valid for constant travel-time and for Euclidean travel-time cells.

Hall, Kamoun, and Sriskandarajah (1997) show that for three-machine cells (problem $RF_3|(free, A, \text{MP}, \text{CRM})|\mu$), the Gilmore and Gomory (1964) algorithm can be used to find the optimal part schedule for the three CRM sequences based on the cycles $S_3 = (A_0, A_1, A_3, A_2)$, $S_4 = (A_0, A_3, A_1, A_2)$, and $S_5 = (A_0, A_2, A_3, A_1)$. The problem is trivial for S_1 because the cycle time does not depend on the part schedule.

Finding the optimal part schedule for the CRM sequences based on the two remaining cycles, $S_2 = (A_0, A_2, A_1, A_3)$ and $S_6 = (A_0, A_3, A_2, A_1)$, is NP-hard, unless special conditions on the data are met (Hall, Kamoun, and Sriskandarajah, 1997). Despite the intractability of the general part scheduling problem for S_2 and S_6, Hall, Kamoun, and Sriskandarajah (1998) develop a polynomial algorithm to find the robot waiting times at the different machines and the cycle time for a *given* part schedule for each of these sequences. Kamoun, Hall, and Sriskandarajah (1999) present heuristics that convert these three-machine problems into a series of two-machine problems, which, as shown in Section 6.2, are easily solvable.

Similar results have been obtained in no-wait robotic cells. For $m = 2$ (problem $RF_2|(no\text{-}wait, E, \text{MP}, \text{CRM})|\mu$), the part scheduling problem for the CRM cycle based on sequence S_2 can be solved using the Gilmore–Gomory algorithm (Agnetis, 2000). For $m = 3$, the part scheduling problem for the CRM cycle based on sequence S_1 is trivial. The part scheduling problems for the CRM cycles based on sequences S_3, S_4, and S_5 can be solved by the Gilmore–Gomory algorithm, and those based on S_2 and S_6 are NP-hard (Agnetis and Pacciarelli, 2000).

Because of the interdependence of the robot's waiting times at the different machines, the expression for the cycle time is often recursive. Therefore, the complexity of the part scheduling problem $RF_m|(free, A, \text{MP}, \text{CRM})|\mu$ increases with the number of nonzero partial waiting times in the cycle. Following this principle, Sriskandarajah, Hall, and Kamoun (1998) develop criteria to assess the complexity of the part scheduling problem in larger ($m \geq 4$) cells. Each cycle is placed into one of four classes based on these criteria, which are, in turn, based on the amount and the locations of the robot's partial waiting in the 1-unit cycle on which the CRM cycle is based, and whether the cell reaches a state E_i^0, for some $i \in \{2, \dots, m\}$. E_i^0 is the state in which all machines except M_i are free, the robot has just completed loading a part onto M_i, and the robot is about to move to the input hopper, I, to perform activity A_0.

Because we are only considering CRM cycles, if a cycle reaches state E_i^0 for some i, then this will be the state of the cell each time a new part enters the cell. Therefore, there are never more than two parts in the cell at any given time. This implies that the time elapsed during the interval between any two consecutive occurrences of E_i^0 can be expressed in terms of the machines' processing times on the two parts in the cell during this interval and a constant that represents the time for the robot's actions (movement, load/unload). Hence, the problem can be analyzed as a traveling salesman problem (TSP): each part is a city, and the distance between two cities is the time of an interval between consecutive occurrences of E_i^0. Specifically, the distance between cities j and k is the time of the interval between the loading of part j onto M_i and the loading of part k onto M_i. Thus, in this case, finding an optimal MPS cycle is equivalent to finding an optimal TSP tour.

Let $\mathcal{N}$ denote the total number of partial waiting times in a 1-unit cycle. Note that there are no 1-unit cycles in which $\mathcal{N} = 1$. The four classes are as follows:

Class U: Cycles in which S_i has no partial waiting ($\mathcal{N} = 0$).
Class V: Cycles with $\mathcal{N} \geq 2$ that reach state E_i^0 at a machine M_i, where $i \in \{2, \dots, m\}$.
Class W: Cycles with $\mathcal{N} \geq 2$ that do not reach state E_i^0 at any machine M_i, where $i \in \{2, \dots, m\}$.

Note that state E_1^0 never occurs in a cycle, and a cycle can belong only to one class. Class V can be further divided into two subclasses:

Class V1: Cycles having $\mathcal{N} = 2$, and at least one partial waiting time occurs at machine M_1 or M_m.
Class V2: All cycles in V other than those in $V1$, i.e., $V2 = V - V1$.

The detailed structure of each problem class is given in Sriskandarajah, Hall, and Kamoun (1998):

Class U: Part schedule independent problems. π_U is the only cycle in this category.
Class V1: Problems that can be solved via the Gilmore and Gomory (1964) algorithm in $O(n \log n)$ time. There are $2m - 3$ such cycles.
Class V2: NP-hard problems that can be formulated as traveling salesman problems (TSP), which allows the use of certain heuristics for TSP. There are

$$\sum_{t=1}^{\lfloor \frac{m}{2} \rfloor} \binom{m}{2t} - 2m + 3.$$

such cycles.
Class W: The remaining problems, which are NP-hard (no TSP structure for these has been found yet). However, those in Class W can be approximated by a heuristic that reduces them to a three-machine problem that uses either S_2 or S_6 (Kamoun, 1994; Kamoun, Hall, and Sriskandarajah, 1999).

7. DUAL GRIPPER ROBOTS

We begin our review of research into dual gripper robots by first considering identical parts. We then look at studies with multiple part-types. Dual gripper robots have been recently studied as a means to improve throughput in cells that are constrained by the robot's speed. These robots can hold two parts simultaneously. In a typical usage of this capability, the robot holds one part while the other gripper is empty; the empty gripper unloads a machine, the robot repositions the second gripper, and it loads that machine. It is generally assumed that the repositioning requires much less time (θ) than does the robot's movement between two adjacent machines (δ) or any machine's processing (p_i). A study that assumes that $\theta = \delta$ can be found in Venkatesh et al. (1997).

When using 1-unit cycles with $\theta = 0$, the throughput of a dual gripper robot cell is equivalent to that of a single gripper robot cell whose machines each have a unitary buffer (Brauner and Finke, 1997; Finke, Gueguen, and Brauner, 1996). Drobouchevitch, Sethi, and Sriskandarajah (2005) demonstrate that, in general, a dual gripper cell can be more productive than a single gripper cell in which each machine has a unitary output buffer. However, they also show that under certain conditions that reflect most real-life cases, the maximum throughput of the dual gripper robot cell can also be achieved by the single gripper robot cell whose machines each have a unitary output buffer.

Having a dual gripper greatly increases the number of feasible cycles. Drobouchevitch, Sethi, and Sriskandarajah (2005) develop a formula to find the number of active cycles in a general cell with m machines. For $m = 2$, there are 52 feasible 1-unit cycles. Among these feasible cycles are 13 that form a dominant subset for problem $RF_m^2|(free, A, cyclic\text{-}1)|\mu$ (Sethi, Sidney, and Sriskandarajah, 2001). In this case, one cycle in particular is optimal under the aforementioned assumption that $\theta \leq \min\{p_i, \delta\}$: S_m^2 exploits the presence of a dual gripper by loading a machine (by switching the grippers) immediately after it finishes unloading a finished part from that machine. The cycle S_m^2 is easy to specify: it starts with the state in which all machines are occupied with parts, and the robot is empty at I. The sequence of activities for the robot in this cycle is given as follows:

```
Cycle $S_M^2$
Begin
    $\epsilon$ : robot unloads a part from $I$
    For $i = 1$ to $m$ do
    Begin
        $\delta$ : robot moves to $M_i$
        $w_i$ : robot waits for the part on $M_i$ to be completed
        $\epsilon$ : robot unloads $M_i$
        $\theta$ : robot switches to the other gripper
        $\epsilon$ : robot loads $M_i$
    End (Next $i$)
    $\delta$ : robot moves to $O$
    $\epsilon$ : robot unloads finished part at $O$
    $\delta$ : robot moves to $I$
End
```

Note: In Drobouchevitch, Sethi, and Sriskandarajah (to appear) and Sethi, Sidney, and Sriskandarajah (2001) it is assumed that the input buffer and the output buffer are in the same location. To achieve consistency with the other models in this paper while maintaining the flavor of these results for cells with dual gripper robots, we have modified this assumption so that these buffers are in separate locations but adjacent, i.e., the machines are arranged in a circle. The cycle time for S_m^2, if identical parts are processed, can easily be calculated (Geismar, Dawande, and Sriskandarajah, 2004b) as

$$T_{\mathrm{a}}\left(S_m^2\right) = \max\left\{(m+2)\delta + 2(m+1)\epsilon + m\theta,\ \max_{1\le j\le m}\{p_j\} + 2\epsilon + \theta\right\}.$$

The best possible improvement achieved by implementing a dual gripper robot is to reduce the cycle time by half (Sethi, Sidney, and Sriskandarajah, 2001). A more common result in an additive travel-time cell producing a single part-type is a reduction by 25–33% (Su and Chen, 1996). Conditions that indicate a possible benefit from the use of a dual gripper robot include

1. m is not large and $\frac{\max p_i}{(\delta+2\epsilon)}$ is large.
2. m is large and $\frac{\max p_i}{(\delta+2\epsilon)}$ is not large.
3. $\frac{\epsilon}{\delta} \le 1$.

For the problem of multiple part-types in a dual gripper robotic cell, we again consider CRM sequences based on 1-unit cycles. For $m = 2$ (problem $RF_2^2|(free, A, \mathrm{MP}, \mathrm{CRM})|\mu$), there are 52 CRM sequences, 13 of which form a dominant subset. Sriskandarajah et al. (2004) demonstrate that the part scheduling problem is NP-hard for 6 of the 13 undominated CRM sequences and is polynomially solvable for the other 7. They also provide computational results that suggest that the productivity gains from adding a dual gripper robot to a cell producing multiple part-types can be as large as 36%. Drobouchevitch et al. (2004) develop a heuristic based on Gilmore and Gomory (1964) that provides a $\frac{3}{2}$-approximation of the optimum for the six NP-hard CRM sequences.

A detailed structural analysis of constant travel-time dual gripper robotic cells with identical parts (problem $RF_m^2|(free, C, cyclic\text{-}1|\mu)$ has been recently done by Geismar, Dawande, and Sriskandarajah (2004b). For simple robotic cells, they prove that S_m^2 has the same cycle time in a constant travel-time cell as it does in a circular additive travel-time cell, and that it is an optimal solution for the constant travel-time 1-unit cycle problem under a condition that is common in practice:

Theorem 6. Assume $\theta \le \min\{\delta, p_1, \ldots, p_m\}$. *A lower bound for cycle times for problem* $RF_m^2|(free, C, cyclic\text{-}1)|\mu$ *is given by*

$$\mathrm{LB} = \max\left\{(m+2)\delta + 2(m+1)\epsilon + m\theta,\ \max_{1\le i\le m}\{p_i\} + 2\epsilon + \theta\right\}.$$

If $\theta \le \delta \le \min\{p_1, \ldots, p_m\}$, *then this lower bound applies to problem* $RF_m^2|(free, C, cyclic\text{-}k)|\mu$.

Corollary 2. For problem $RF_m^2|(free, C, cyclic\text{-}1)|\mu$, *cycle* S_m^2 *is optimal among all one-unit cyclic schedules under the assumption that* $\theta \le \min\{\delta, p_1, \ldots, p_m\}$. *For problem* $RF_m^2|(free, C, cyclic\text{-}k)|\mu$, *cycle* S_m^2 *is optimal among all cyclic schedules under the assumption that* $\theta \le \delta \le \min\{p_1, \ldots, p_m\}$.

Additionally, computational results indicate that the productivity gains from adding a dual gripper robot to a constant travel-time cell that produces a single part-type are similar to those of a corresponding additive travel-time cell.

Theorem 6 remains valid for additive travel-time dual gripper cells if all the machines M_i, $i = 0, 1, \ldots, m, m+1$, are placed equidistant around a circle, and the travel time between two adjacent machines M_i and M_{i+1}, $i = 0, 1, \ldots, m, m+1$, is a constant δ, where we assume $m + 2 \equiv 0$. The results for constant travel-time dual-gripper cells with parallel machines are provided in the next section.

8. PARALLEL MACHINES

In the classical parallel machine part-scheduling problem, jobs are processed by identical machines in parallel, but the system is not a flowshop. Each job requires only a single operation, and it may be processed on any of those machines (Pinedo, 1995). Hall, Potts, and Sriskandarajah (2000) analyze such systems in which all jobs must be loaded (setup) by a common server. They provide either polynomial or pseudo-polynomial algorithms, or a proof of NP-completeness for various conditions on setup times, processing times, and objectives. Błażewicz et al. (1991) analyze the Vehicle Routing with Time Windows problem, in addition to the part-scheduling problem, for a similar system of parallel machines, each of which can perform various tasks. These machines are served by several automated guided vehicles that travel the same circuit.

8.1. Single gripper robots

We now examine the use of parallel machines in robotic flowshop cells for cells producing identical parts (problem $RF_m(m_1, m_2, \ldots, m_m)|(\mathit{free}, C, \mathit{cyclic}\text{-}k)|\mu$). In certain cells, throughput can be improved by adding an identical machine to a particular processing stage. Such a machine would be used in parallel with the other machine of that stage. This method is especially cost effective if there are a small number of machines whose processing times are significantly larger than those of the other machines. In fact, using m_j parallel machines at stage j reduces that stage's impact on the per unit cycle time's lower bound by a factor of m_j: $T_c(\pi)/k \geq (p_j + 3\delta + 4\epsilon)/m_j$, where cycle π produces k parts (Geismar, Dawande, and Sriskandarajah, 2004a). Herrmann et al. (2000) devise a network model that can be used to perform sensitivity analysis to determine by how much the addition of a parallel machine to a specific stage reduces the cycle time (see Section 10.2.1).

8.1.1. Definitions

Just as a simple robotic cell is analogous to a flowshop with blocking, a robotic cell with parallel machines is analogous to a flexible flowshop with blocking. In a robotic cell with parallel machines, there are m stages, and for each processing stage i there are $m_i \geq 1$ identical machines. As with simple cells, each part is processed at each stage according to the same fixed sequence. A part can be processed at stage i by any one of the m_i machines at that stage.

The m_i distinct machines at stage i are denoted M_{ia}, M_{ib}, M_{ic}, etc. Each machine at stage i has processing time p_i. For a constant travel-time robotic cell, $d(M_{i\gamma}, M_{j\eta}) = \delta$, if $i \neq j$ or $\gamma \neq \eta$, whether the robot is carrying a part or not.

For clarity and flexibility, we define the concept of *activity* for a system of parallel machines. When transferring a part from one machine to another, the activity is denoted with three subscripts, e.g., $A_{i\gamma\eta}$. This indicates that a part is being transfered from stage i to stage $i + 1$, is being unloaded from machine $M_{i\gamma}$, and is being loaded onto machine $M_{i+1,\eta}$. If there is only a single machine at the source or destination stage, instead of a letter, use the asterisk symbol (*). For example, activity A_{1ba} means that the robot takes the part from M_{1b}, travels to M_{2a}, and loads the part onto M_{2a}. To signify taking a part from I, moving to M_{1a}, and loading M_{1a}, we write A_{0*a}.

8.1.2. LCM *cycles*

In order to use parallel machines effectively, the robot must execute a multiunit cycle. Otherwise, its throughput would be no better than that of a simple cell. As we have previously seen (Section 5), when dealing with multiunit cycles, one must address feasibility. To avoid infeasible cycles, Kumar, Ramanan and Sriskandarajah (2005) use LCM *cycles* in their genetic algorithm-based analysis of a specific company's robotic cell. They also show that LCM cycles greatly increase throughput.

LCM cycles are so named because one cycle produces a number of units equal to the least common multiple, denoted λ, of the numbers of machines ($m_i, i \in M$) in each stage. This allows each machine of a specified stage to be used an equal number of times during a cycle.

LCM cycles are a special class of multiunit cycles called *blocked cycles*. A k-unit blocked cycle contains k blocks, each with $m+1$ activities—one for each stage. In each block, the order of the activities by number, called the *base permutation*, is the same for a specific cycle. The letters that specify the machines to be unloaded and loaded for specific stages change from block to block. Here is a blocked cycle based on the permutation (0, 1, 2) with $k = 6$:

Example.

$$\pi = A_{0*a}A_{1bc}A_{2a*} \quad A_{0*b}A_{1ab}A_{2c*} \quad A_{0*a}A_{1bc}A_{2b*} \quad A_{0*b}A_{1aa}A_{2c*} \quad A_{0*a}A_{1bb}A_{2a*} \quad A_{0*b}A_{1aa}A_{2b*}$$

Definition. LCM *Cycles* are blocked cycles that have the following characteristics:

- Each machine is loaded as soon as possible after it is unloaded, as allowed by the base permutation.
- For each stage, each of its machines has the same number of activities between its loading and its unloading each time it is used.
- The cycle produces $\lambda = \text{LCM}[m_1, m_2, \ldots, m_m]$ parts.
- For each stage, the loading of its machines is ordered alphabetically, beginning with machine a in the first block.

As a result of the first two conditions, for each stage $i \in M$, each of its machines is used λ/m_i times per cycle. These two conditions also imply that the cycle is feasible. The last requirement ensures that a cell has a unique LCM cycle for a given base permutation.

Geismar, Dawande, and Sriskandarajah (2004a) prove that LCM cycles form a dominant subset of blocked cycles. Furthermore, for the very common practical case in which $p_i \geq \delta, \forall i \in M$, π_{LD}, which is the LCM cycle based on the reverse cycle π_{D}, is optimal over all cycles. For the two-machine cycle with $m_1 = 2$ and $m_2 = 3$, the reverse LCM cycle is

$$\pi_{\text{LD}}(2,3) = A_{0*a}A_{2a*}A_{1ba} \quad A_{0*b}A_{2b*}A_{1ab} \quad A_{0*a}A_{2c*}A_{1bc} \quad A_{0*b}A_{2a*}A_{1aa} \quad A_{0*a}A_{2b*}A_{1bb} \quad A_{0*b}A_{2c*}A_{1ac}$$

The cycle time is $T(\pi_{\text{LD}}(2,3)) = \max\{36\delta + 36\epsilon, 3(p_1 + 3\delta + 4\epsilon), 2(p_2 + 3\delta + 4\epsilon)\}$. The cycle time for the general reverse LCM cycle is

$$T(\pi_{\text{LD}}) = \max\left\{2\lambda(m+1)(\delta+\epsilon),\ \max_{1\leq i\leq m}\left\{\frac{\lambda}{m_i}(p_i + 3\delta + 4\epsilon)\right\}\right\}. \tag{4}$$

In much the same way that π_{D} provides a 2-approximation for simple cells (Section 5.3), π_{LD} is a 2-approximation for robotic cells with parallel machines (problem $RF_m(m_1, \ldots, m_m)$ | $(free, C, cyclic\text{-}k)|\mu$), regardless of the machines' processing times' relationships to δ (Geismar, Dawande, and Sriskandarajah, 2004a). We know of no studies that produce better approximations for the per unit cycle time for robotic cells with parallel machines.

If the robot's moves are sequenced by using π_{LD}, equation (4) can be used to determine how many parallel machines are required for each stage in order to meet a specified throughput requirement (Geismar, Dawande, and Sriskandarajah, 2004a). Suppose that the average per unit cycle time must be less than T^*. Thus, $(p_i + 3\delta + 4\epsilon)/m_i \leq T^*, \forall i$ (if $2(m + 1)(\delta + \epsilon) > T^*$, this time requirement cannot be satisfied), so $(p_i + 3\delta + 4\epsilon)/T^* \leq m_i$. Therefore, as in Geismar, Dawande, and Sriskandarajah (2004a), we have

$$m_i = \left\lceil \frac{p_i + 3\delta + 4\epsilon}{T^*} \right\rceil, \quad i = 1, \ldots, m.$$

It is interesting to note that although $T(\pi_{\text{LD}})/\lambda$ is the average time to produce one unit, finished parts are not necessarily completed at a constant rate. Rather, the times between the output of successive parts can be periodic, based on processing times of the parallel machines and the shorter processing times of the single machines. Perkinson, Gyurcsik, and McLarty (1996) provide an example of a three-stage cell running cycle π_{LD} in which $p_1 = p_2$ are short, and p_3 is very long with $m_1 = m_2 = 1$ and $m_3 = 3$. In this case, the differences in the loading times between M_{3a} and M_{3b} and between M_{3b} and M_{3c} are each the relatively small quantity $p_2 + \delta + 2\epsilon$. However, M_{3a} is loaded almost p_3 time units after M_{3c}. Thus, the unloading of parts is periodic: all three are unloaded within an interval of size $2(p_2 + \delta + 2\epsilon)$, but there is a gap of almost p_3 time units until M_{3a} is unloaded again.

8.2. Dual gripper robots

For dual gripper robotic cells with parallel machines, the only analysis available is that for constant travel-time cells (problem $RF^2_m(m_1, \ldots, m_m)|(free, C, cyclic\text{-}k)|\mu$) by Geismar, Dawande, and Sriskandarajah (2004b). They consider an m-stage cell with $m_i \geq 1$ machines for stage i, $i = 1, \ldots, m$. Under conditions that are common in practice, they provide an optimal solution to the k-unit cycle problem. The main idea of this analysis is the construction of a specific cycle, $S^2_{m,L}$, which combines the structures of LCM cycles (Section 8.1.2) and the cycle S^2_m for dual gripper simple cells (Section 7). We avoid providing the exact description of the cycle $S^2_{m,L}$; instead, we offer its characteristics to help develop intuition:

- The cycle begins with all machines occupied and the robot at the input buffer while holding no part.
- The number of parts produced in one cycle is $\lambda = \text{LCM}[m_1, m_2, \ldots, m_m]$, the least common multiple of the number of machines in each stage.
- For each stage i, the unloading of its machines is ordered alphabetically, beginning with machine M_{ia} in the first block.
- The dual gripper is used effectively: each machine is loaded immediately after it is unloaded.
- After loading a machine in stage i, the robot travels to stage $i+1$, $i = 1, \ldots, m$. If $1 \leq i \leq m-1$, then the robot unloads and then loads some particular machine in stage $i+1$. If $i = m$,

the robot places the part into the output buffer (O), travels to the input buffer (I), and collects a new part that it carries to a machine in stage 1.

- For each stage *i*, each of its machines is used $\frac{\lambda}{m_i}$ times.

Under the practically common condition $\theta \leq \delta \leq \min\{p_1, \ldots, p_m\}$ (see Section 7), the following result underlines the importance of the cycle $S^2_{m,L}$.

Theorem 7. In a robotic cell with parallel machines and a dual gripper robot (*problem* $RF^2_m(m_1, \ldots, m_m)|(free, C, cyclic\text{-}k)|\mu)$, $S^2_{m,L}$ *is optimal among all k-unit cyclic schedules* ($k \geq 1$) *under the assumption that* $\theta \leq \delta$ *and* $p_i \geq \delta, i = 1, \ldots, m$.

In practice, the main motivation for using either a dual gripper robot or a parallel machine is to increase productivity of the robotic cell. Therefore, the quantification of the productivity gains from using a dual gripper (instead of a single gripper) robot and parallel machines has important practical implications. Geismar, Dawande, and Sriskandarajah (2004b) consider three ways in which production managers can hope to increase productivity (keeping other process-related parameters the same): (i) use of a dual gripper robot, (ii) use of parallel machines, and (iii) use of both a dual gripper robot and parallel machines. They obtain analytical expressions for these gains and demonstrate significant improvements on real-world cells used at a semiconductor equipment manufacturer, e.g., given a robotic cell with parallel machines, adding a dual gripper robot can improve throughput by 40%.

9. MULTIPLE ROBOTS

Manufacturers employ additional robots in a cell in order to increase throughput by increasing the material-handling capacity. The intent is for the robots to work concurrently and without hindering each other. Additional robots may also provide flexibility when designing the layout of a cell, especially for manufacturing processes that require a large number of stages and, hence, more space.

Two early studies of systems with multiple robots focused on cells that are not flowshops. The robots move to different workstations to perform a variety of assembly functions for different parts by using different tools. Maimon and Nof (1985) discuss enumerative algorithms that determine how high-level decisions such as work allocation, cooperation, and the robots' routes of travel are made by the control computer during operations. Nof and Hannah (1989) analyze how productivity is effected by different types of cooperation among the robots and by the level of resource sharing by the robots.

Geismar, Sriskandarajah, and Ramanan (2004d) studied a multiple robot flowshop cell with parallel machines that was designed and developed by a semiconductor equipment manufacturer. The study proposes a simple plan of operation that allows the robots to operate concurrently and with no risk of colliding. They propose sequences of robots' moves that are a generalization of π_{LD} (see Section 8). This scheme is optimal in constant travel-time cells (problem $RF_{m,r}(m_1, m_2, \ldots, m_m)|(free, E, cyclic\text{-}k)|\mu)$ if $p_i \geq \delta, \forall i$. Additionally, circumstances under which it is optimal for Euclidean travel-time cells (problem $RF_{m,r}(m_1, m_2, \ldots, m_m)|(free, E, cyclic\text{-}k)|\mu)$ are provided. A computational study demonstrates this scheme's superiority to the one currently used by the manufacturer.

Results for multiple robot no-wait cells (problem $RF_{m,r}|(no\text{-}wait, E, cyclic\text{-}k)|\mu$) have been derived in Kats (1982) and Kats and Levner (1997, 1998). These are summarized in Crama et al. (2000).

10. IMPLEMENTATION ISSUES

We now examine issues that have been addressed in the practice-oriented literature: time required to reach steady state, computing a lot's makespan, local material handling devices, and revisiting machines. Less theoretical work has been done on these topics, so there are fewer general results.

10.1. Reaching steady state

Since most results are derived for systems in steady state (defined in Section 5), it is natural to investigate how much processing is required in order to reach steady state from an empty cell. For $m = 3$, a steady state for cells that process multiple parts (problem $RF_3|(free, A, \text{MP}, \text{CRM})|\mu$) can be obtained from an empty cell by CRM sequences based on cycles S_1, S_3, S_4, and S_5 during the production of the first MPS (CRM and MPS are defined in Section 6). For cycle S_1, this is trivial. For CRM sequences based on cycles S_3, S_4, and S_5, this occurs because while executing these sequences, the cell reaches state E_2^0 or E_3^0 (E_1^0 is defined in Section 6.3). Because machine M_i begins its processing and all other machines are empty at state E_i^0, this state effectively decomposes production into a single MPS. This can be generalized to larger cells: for any m-machine cell, any cycle that places the cell into a state E_i^0 for some $i \in \{2, \ldots, m\}$ reaches steady state while processing the first MPS (Hall, Kamoun, and Sriskandarajah, 1997).

Hall, Kamoun, and Sriskandarajah (1997) also show that it may take longer for the two other three-machine cycles to reach steady state. For S_2 and S_6, the cell converges to a steady state in a number of cycles which is bounded by a function of the cell data. Specifically, the CRM sequence based on cycle S_2 goes through at most $\max\{1, \min\{s_i\}\}$ cycles before reaching steady state, where

$$s_i = p_{3,\sigma(i)} - \max\left\{\beta_2, p_{1,\sigma(i+1)} - \max\left\{0, p_{2,\sigma(i+1)} - \beta_2\right\}\right\}, \quad i = 1, 2, 3,$$

$$\beta_2 = \sum_{i=0}^{3} \epsilon_i^u + \sum_{i=1}^{4} \epsilon_i^l + 2\sum_{i=1}^{4} \delta_i - 3\eta,$$

and σ is the order in which the part-types are processed. The result is the same for the CRM sequence based on cycle S_6, except

$$s_i = p_{1,\sigma(i)} - \max\left\{\beta_6, p_{2,\sigma(i-1)}\right\}, \quad i = 1, 2, 3,$$

$$\beta_6 = \sum_{i=0}^{3} \epsilon_i^u + \sum_{i=1}^{4} \epsilon_i^l + 2\delta_1 + 4\delta_2 + 4\delta_3 + 2\delta_4 - 4\eta.$$

10.2. Calculating the makespan of a lot

We now present three methods for calculating the makespan of a lot. One is graphical, the other two are algebraic.

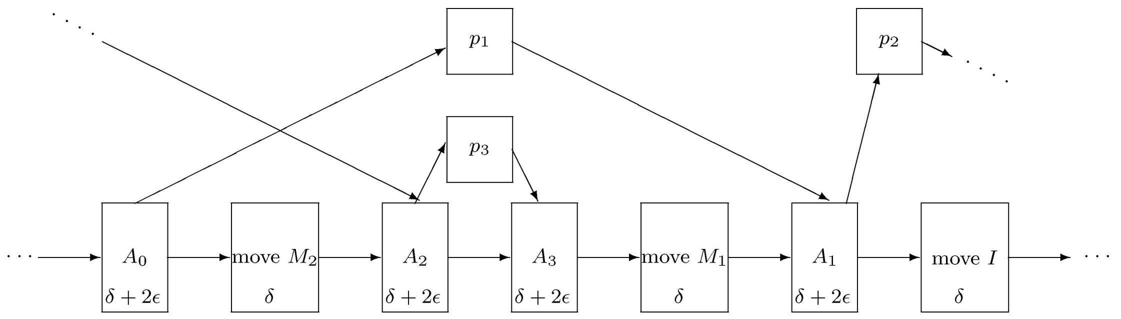

Figure 3. Portion of graph to calculate makespan of a lot using cycle (A_0, A_2, A_3, A_1).

10.2.1. Graphical approach

Herrmann et al. (2000) use a directed acyclic graph to calculate a lot's makespan. Each node represents either the robot's movement or a machine's processing of a part. A node is labeled with the time it requires. Arcs indicate precedence constraints: $(j, k) \in A$ if action j must precede action k. For example, there is an arc originating at the node representing activity A_1 (labeled $\delta + 2\epsilon$) whose destination is the node representing processing on M_2 (labeled p_2). In total, there are l nodes for each activity, where l is the lot size. The total time needed to process a lot equals the length of the network's longest path, called the *critical path*. This path is found via a dynamic programming algorithm. An example of a portion of such a graph for cycle (A_0, A_2, A_3, A_1) is in Figure 3.

This formulation is very useful for sensitivity analysis. The effect of an increase in the time of an operation (either a processing time or a movement time) depends on the size of the increase and whether that operation is on the critical path. If it is not on the critical path, an increase may have no effect on the makespan. However, a large enough increase may put the operation onto the critical path. Thus, the effect of changing one activity is a nondecreasing piecewise linear function.

10.2.2. Algebraic approaches

Wood (1996) calculates a semiconductor lot's total processing time by using two parameters. The first is the incremental lot cycle time (t), which is defined as the average increase in the lot cycle time that results from a lot size increase of one wafer. The fixed time (T) represents the lot setup time that is independent of the lot size. If the lot size is l, then a lot's total processing time is $CT = T + lt$.

The cell's throughput is often improved by using multiple input devices, which are called *cassette ports* in this implementation. If there are n_L such ports and each holds a lot, then, by Little's (Little, 1961) formula, the throughput is

$$\mu = \frac{n_L l}{T + lt}.$$

This assumes that the cell's performance is constrained by the input supply. If not, then $\mu = 1/t$ (Wood, 1996).

The previous two methods find a lot's total processing time by assuming that the cell is in steady state throughout its entire processing. We now examine a more realistic model by Perkinson et al. (1994) that accounts for the times that the cell is not in steady state.

A lot's total processing includes five stages: loading the parts to be processed into the robotic cell (T_{load}), transition from start into steady state (T_{A}), steady state (T_{S}), processing the final parts (T_{B}), and unloading the completed parts (T_{unload}). Obviously, the expression for the total lot makespan is $T_{\text{L}} = T_{\text{load}} + T_{\text{A}} + T_{\text{S}} + T_{\text{B}} + T_{\text{unload}}$.

T_{load} and T_{unload} are given constants.

$T_{\text{S}} = \text{FP}(l - m + 1)$, where l is the lot size and FP is the fundamental period, i.e., the steady state per unit cycle time.

The times for the transition periods are

$$T_{\text{A}} = z(p + 2\delta) + \sum_{i=z+1}^{m-1} 2i\delta$$

$$T_{\text{B}} = z(p + 4\delta) + \sum_{i=z+1}^{m-1} 2(i + 1)\delta - 3\delta,$$

where p is the processing time for each machine, δ is the travel-time between any two machines ($\epsilon = 0$ in this model), and $z = \min\{m - 1, \lfloor \frac{p/\delta+2}{2} \rfloor\}$. Moreover, z represents the maximum number of machines that can be in use before the robot is always busy.

The expressions for T_{A}, T_{B}, and T_{S} indicate that the makespan of a lot is minimized by minimizing the fundamental period, which is done by the same methods as those previously discussed for maximizing throughput.

10.3. Local material handling devices

Local material handling devices (LMHD) can be used to increase throughput if the robot is the bottleneck. An LMHD transports a part from stage j to stage $j + 1$, $j = 1, \ldots, m - 1$, independently of the robot. If LMHDs are used in cells with parallel machines, then $m_j = m_{j+1}$, and each machine of stage j is linked to a unique machine of stage $j + 1$. Furthermore, no stage can be linked to two stages: if stage j is linked to stage $j + 1$, then stage $j - 1$ is not linked to stage j, and stage $j + 1$ is not linked to stage $j + 2$. There are additional physical constraints (e.g., machine sizes, cell layout) that limit the number of links in a particular cell. For ease of exposition, for the remainder of this section we describe only a simple robotic cell.

LMHDs increase throughput by transferring a completed part as soon as possible and by reducing the robot's workload. The robot never performs activity A_j (unload M_j, transport the part to M_{j+1}, load M_{j+1}). In a simple cell with only one LMHD link (from M_j to M_{j+1}), a 1-unit cycle is very similar to a 1-unit cycle for a cell with $m - 1$ machines: it contains m activities $\{A_0, \ldots, A_{j-1}, A_{j+1}, \ldots, A_m\}$, and there are $(m - 1)!$ possible cycles.

When the robot arrives at M_j to load a part, the linked pair of machines is either empty or contains one part. If the robot loads an empty pair, then the cell's operations and the calculations of its waiting times and cycle time are identical to those of an $(m - 1)$-machine cell in which machine M_j^* is loaded and unloaded from different ports, there are $m - 1$ machines, and $p_j^* = p_j + p_{j+1} + y_j$, where y_j is the time required for the LMHD to transfer the part from M_j to M_{j+1}. If the pair has one part when the robot arrives to load M_j, then M_j may still be processing the previous part. In this case, the robot must wait for M_j to complete processing and for the LMHD to transfer that part to M_{j+1}.

Another waiting time occurs if M_j completes processing, only to find M_{j+1} still occupied. In this case, the LMHD cannot transfer the part until the robot unloads M_{j+1}. Thus, a completed part waits on M_j until M_{j+1} is unloaded.

Kumar, Ramanan, and Sriskandarajah (2005) used simulation, linear programming, and genetic algorithms to demonstrate how LMHDs can improve throughput for a particular company's robotic cell. To our knowledge, there has been no general theoretical study to establish an optimal cycle for such an implementation or to determine the conditions which most favor that two machines be joined by an LMHD.

10.4. Revisiting machines

Thus far, we have discussed techniques to improve throughput. We now examine a way to reduce cost without degrading throughput, i.e., to improve the performance-to-cost ratio.

In semiconductor manufacturing, certain stages, e.g., bake and chill, are repeated. If the machine that performs the bake at stage α also performs a bake of the same duration at stage β, an m-machine process can be performed by $m - 1$ machines. This could provide a significant savings in capital investment.

To maintain throughput while reducing the number of machines, the processing time of the revisited machine M_α must be less than one-half of the cell's other processing times: $p_\alpha \leq \min_{i \neq \alpha} p_i/2$. Furthermore, since a revisiting sequence requires more robot movements, δ and ϵ should be relatively small in comparison to $\max_{1 \leq i \leq m} p_i$, e.g., $\max_{1 \leq i \leq m} p_i + 3\delta + 4\epsilon \geq 2(m+1)(\delta + \epsilon)$ in a constant travel-time cell.

Perkinson, Gyurcsik, and McLarty (1996) implement such a revisitation scheme in a three-machine cell that is used for a four-stage process (problem $RF_4 \mid (\mathit{free}, C, \mathit{cyclic}\text{-}1) \mid \mu$). The load lock (combined Input/Output device) is used as a buffer to store partially completed parts before their second visit to M_1 for processing stage 4. Their scheme, shown in Figure 4, is based on π_D (each machine is reloaded as soon as possible after it is unloaded) and has cycle time

$$T_c(\pi_1) = \max\{p_1 + 10\delta + 12\epsilon, 2p_1 + 5\delta + 8\epsilon, p_2 + 3\delta + 4\epsilon, p_3 + 3\delta + 4\epsilon\}.$$

A particular implementation may not allow the use of the load lock as a buffer. It is still possible to revisit a machine in this case. An example of such a sequence is shown in Figure 5. Its cycle time is

$$T_c(\pi_2) = \max\left\{p_1 + 8\delta + 10\epsilon, 2p_1 + 5\delta + 8\epsilon, p_1 + p_2 + 5\delta + 8\epsilon, p_3 + 3\delta + 4\epsilon, \frac{2p_1 + p_2 + p_3 + 5\delta + 10\epsilon}{2}\right\}.$$

To our knowledge, there has been no thorough study of revisitation schemes to determine general guidelines or to evaluate the tradeoffs between performance and cost.

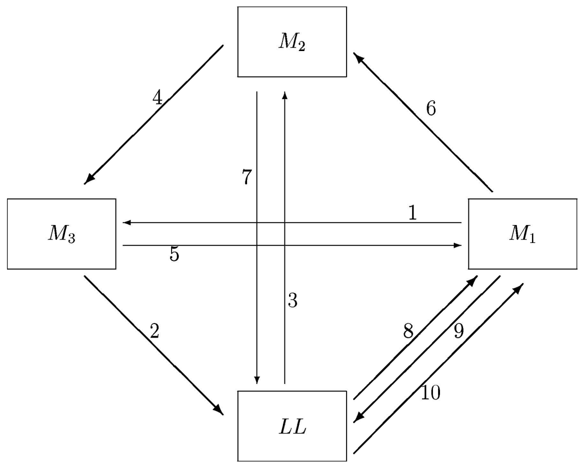

Figure 4. Machine revisitation sequence using load lock for intermediate storage. numbers correspond to the order of robot moves. Arcs 2, 4, 6, 8, 9, and 10 represent robot moves with a part.

11. ROBOTIC CELLS IN PRACTICE

Robotic cells often perform the general functions of arc welding, material handling, and machining (Asia Pacific Metal Working, 2000). In addition to semiconductor manufacturing, electroplating, and textiles, they are used in many different industries, including injection molding of battery components (Vulcan Publications, Inc., 2001), glass manufacturing and processing (Glass on Web, 2002), building products (Products Finishing Magazine, 2001), cosmetics (Rapid Development Services, 2001), lawn tractors (Remich, 2000), and fiber-optics (Zmation, Inc., 2002). In the medical field, robotic cells are used to produce components for magnetic resonance imaging systems (Manufacturing Talk, 2001), for automated pharmacy compounding (RoboDesign International, Inc., 2002), to process nucleic acids, and to generate compounds for tests in relevant biological screens (Rudge, 1997). Cells for grinding, polishing, and buffing handle many products, including rotors, stainless steel elbows for the chemical and the food industries, sink levers and faucets, propane tanks, flatware, and automotive products (Sawyer and Smith, 2001).

In modern manufacturing, a typical robot may move along six axes (including linear translation) and have a three-fingered pneumatic gripper (Manufacturing Talk, 2001). Some have angular and parallel motion grippers that include miniature, low profile, sealed, long jaw travel and 180 degree jaw motion grippers (Applied Robotics, Inc., 2001). Robots that can calculate the optimal path between two locations or that can quickly change their tools are common (Asia Pacific Metal Working, 2000). Robotic vision-guided systems have grown in the market, especially for assembly cells (Robot Workspace Technologies, Inc., 1999).

The economic benefits of robotic cells extend beyond increasing the efficiency of manufacturing. One company states that its 19 cells will achieve their payoff mark in only 2 years (Vulcan Publications, Inc., 2001). Another notes that implementing robotic cells has consolidated several processes, which has reduced floor space requirements (Products Finishing Magazine, 2001). Such successes have prompted the robotic cell market to grow at approximately 15% annually for the past few years (Asia Pacific Metal Working, 2000).

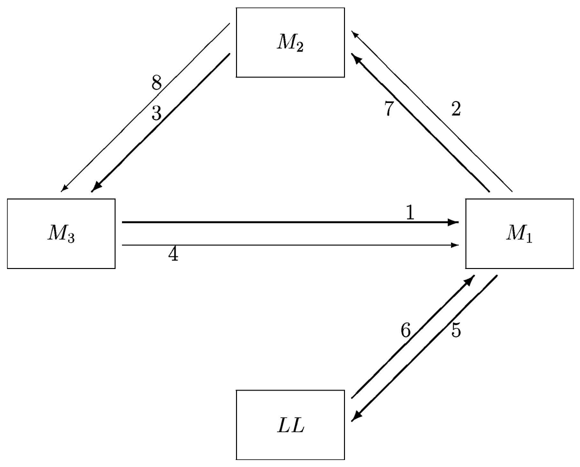

Figure 5. Machine revisitation sequence without using load lock for intermediate storage. Numbers correspond to the order of robot moves. Arcs 1, 3, 5, 6, and 7 represent robot moves with a part.

Companies use simulators to study their robotic cells because semiconductor factories are often too large, too complex, and too costly to optimize and refine any easier way. There are currently no analytical models that accurately depict cells with general travel times, stochastic processing times, and random machine failures. Some simulators claim to model an automated system with better than 98% accuracy (Medeiros et al., 1998).

Among the topics studied via simulators are the influence of adding a parallel machine to a bottleneck process, the effects of equipment failures and maintenance on performance, and the disruption caused by introducing high-priority jobs into steady state production lines (Duenyas, Fowler, and Schruben, 1994). Simulators are also used for sensitivity analysis to determine if equipment purchases are required to meet new production goals (Medeiros et al., 1998). Because simulators can predict and verify cycle times before a cell is assembled, they can be used to improve the cell's layout during the planning stage. This capability can also influence product design by leading to changes that make the product's fabrication more efficient in an existing cell (Smart Media Group Ltd, 2001).

For cells with stochastic data, some companies implement feedback control by using dispatching rules that determine the robot's next move based on the cell's current state. One such rule is called Longest Waiting Pair. To implement this scheme, the control computer tracks each part whose processing has completed on its current machine and is waiting to be moved. It also tracks each empty machine that is waiting for the next part to process. Each part's waiting time is summed with the waiting time of the machine to which it next travels. For the pair with the largest combined waiting time, the robot's next move is to carry the part to its corresponding machine (Kumar, Ramanan, and Sriskandarajah, 2005).

When designing a cell and its operating parameters, the main objective is the maximization of the cell's throughput rate. Intermediate goals toward reaching this objective are high machine utilization and a smooth distribution of work over the entire system. Management must balance the pursuit of these goals with its desire to reduce work-in-process inventory levels. As the system operates, bottleneck identification and knowing which lots might be late become important objectives (Duenyas, Fowler, and Schruben, 1994). General guidelines for applying operations research techniques to planning, designing, and operating robotic cells can be found in Hall et al. (1999).

12. OPEN PROBLEMS: RECOMMENDATIONS FOR FUTURE STUDY

We now summarize some open problems in the field of robotic cell sequencing and scheduling.

- *Simple robotic cells*: As mentioned in Section 3, at least nine different types of simple robotic cells with single gripper robots have been studied in the literature, depending on the pickup criterion (no-wait, interval, or free-pickup) and the travel-time metric (additive, constant, or Euclidean). For additive and constant travel-time cells with free-pickup, there exist polynomial-time algorithms for finding an optimal 1-unit cycle (Crama and van de Klundert, 1997a; Dawande, Sriskandarajah and Sethi, 2002) (problems $RF_m|(free, A, cyclic\text{-}1)|\mu$ and $RF_m|(free, C, cyclic\text{-}1|\mu)$. For such cells, two fundamental problems remain open: (1) For a given $k, k \geq 2$, finding an algorithm that produces an optimal k-unit cycle and (2) Finding an algorithm that produces an optimal cycle among the class of all cyclic solutions (problems $RF_m|(free, A, cyclic\text{-}k)|\mu$ and $RF_m|(free, C, cyclic\text{-}k)|\mu$). The complexity status of these problems is unknown. It is conceivable that mathematical programming formulations may be of help in solving these problems. To the best of our knowledge, there is no known linear or integer programming formulation for the optimal k-unit cycle problem, $k \geq 1$, for any cells with free-pickup.
  For interval robotic cells with additive travel-times (problem $RF_m|(interval, A, cyclic\text{-}1)|\mu$), the robot move sequencing problem is NP-hard (Crama and van de Klundert, 1997b). This implies that the problem is NP-hard for interval cells with Euclidean travel-times, too (problem $RF_m|(interval, E, cyclic\text{-}1)|\mu$). Consequently, approximation algorithms that guarantee a near-optimal solution in polynomial time are of interest for such cells. No results have been published for interval cells with constant travel-time.
  For Euclidean no-wait cells, polynomial algorithms to find an optimal 1-unit cycle (Levner, Kats, and Levit, 1997) (problem $RF_m|(no\text{-}wait, E, cyclic\text{-}1)|\mu$) and an optimal 2-unit cycle (Che, Chu, and Levner, 2003) (problem $RF_m|(\text{no-}wait, E, cyclic\text{-}2)|\mu$) have been developed. Finding an optimal k-unit cycle over all $k, k \geq 1$, (problem $RF_m|(no\text{-}wait, E, cyclic\text{-}k)|\mu$) is an open problem, although Agnetis (Agnetis, 2000) has conjectured that $k < m$ for such an optimal cycle. Mangione, Brauner, and Penz (2003) prove this conjecture for certain balanced ($p_i = p, \forall i$) and regular additive no-wait cells.
- *Simple robotic cells with multiple part-types*: We saw in Section 6 that for multiple part-type cells, two subproblems arise: part scheduling and robot move sequencing. Given a robot move sequence, finding the optimal part schedule is NP-hard for $m \geq 3$ (Hall, Kamoun, and Sriskandarajah, 1997). The complementary problem of finding an optimal robot move sequence given a part schedule has not been studied for $m \geq 3$.
  For cells that process multiple part-types with $m \geq 3$, all results in Section 6.3 concern CRM sequences (problem $RF_m|(free, A, \text{MP}, \text{CRM})|\mu$). There has been no research on finding the optimal part schedule for a non-CRM MPS robot move sequence (problem $RF_m|(free, A, MP, cyclic\text{-}k)|\mu$). Such a result would provide insight into how close the best CRM sequence is to the optimal MPS cyclic solution. This problem, however, is a generalization of the optimal k-unit cycle problem, which is also open.

- *Robotic cells with parallel machines*: The optimal cyclic solution has been found only for a special case ($p_i \geq \delta, \forall i$) in free-pickup constant travel-time cells (see Section 8 or Geismar, Dawande, and Sriskandarajah (2004a)). No studies have been published that address more general free-pickup constant travel-time cells (problem $RF_m(m_1, m_2, \ldots, m_m)|(\textit{free}, C, \textit{cyclic-k})|\mu$) or any free-pickup cells with additive travel-times (problem $RF_m(m_1, m_2, \ldots, m_m)|(\textit{free}, A, \textit{cyclic-k})|\mu$).
  For cases in which finding the optimal move sequence in a simple robotic cell is known to be NP-hard (Euclidean free-pickup: $RF_m(m_1, m_2, \ldots, m_m)|(\textit{free}, E, \textit{cyclic-k})|\mu$, Euclidean interval: $RF_m(m_1, m_2, \ldots, m_m)|(\textit{interval}, E, \textit{cyclic-k})|\mu$, and additive interval: $RF_m(m_1, m_2, \ldots, m_m)|(\textit{interval}, E, \textit{cyclic-k})|\mu$), the problems for cells with parallel machines are NP-hard, too. Finding efficient approximation algorithms remains an open question for these cases.
  There exists a polynomial algorithm to optimize throughput for 1-unit cycles in no-wait simple robotic cells (Levner, Kats, and Levit, 1997). We know of no work that addresses no-wait robotic cells with parallel machines (problem $RF_m(m_1, m_2, \ldots, m_m)|(\textit{no-wait}, ^*, \textit{cyclic-k})|\mu$).
  Finding the optimal schedule for processing multiple part-types in robotic cells with parallel machines (for any travel-time or pickup scheme (problem $RF_m(m_1, m_2, \ldots, m_m)|(^*, ^*, \mathrm{MP}, \textit{cyclic-k})|\mu$)) is also a generalization of the optimal k-unit cycle problem. Hence, it is currently an open question.
- *Stochastic data*: In general, one cannot expect deterministic solutions to be optimal for stochastic problems. It is important to find conditions under which deterministic solutions provide near-optimal solutions. Finding the best dispatching rule is also an open problem. Another issue concerns random failures of processing machines. Some manufacturers use parallel machines to provide redundancy. Given each machine's rate of failure, the distribution of each machine's repair time, each machine's cost, and the distribution of each machine's processing time, what is the economically optimal number of machines to have at each stage? Such a calculation should also formulate the economic impact of improved throughput, i.e., cycle time comparison is a subproblem to the problem of finding the optimal number of redundant machines. Suri (1985) provides some guidelines for addressing such questions.
- *Dual gripper robots*: These have not been studied for Euclidean travel-time (problem $RF_m^2|(^*, E, \textit{cyclic-k})|\mu$) or for interval or no-wait pickup (problems $RF_m^2|(\textit{interval}, ^*, \textit{cyclic-k})|\mu$ and $RF_m^2|(\textit{no-wait}, ^*, \textit{cyclic-k})|\mu$). It would also be interesting to examine the use of dual gripper robots in cells with multiple robots and parallel machines. Such a study could be based on the results in Geismar, Sriskandarajah, and Ramanan (2004d).
- *Flexible robotic cells*: Recent work has studied robotic cells that are open shops, rather than flowshops. In these cells the operations can be performed in any order, and each machine can be configured to perform any of the operations. Geismar et al. (2004c) show that for $m = 3$ and $m = 4$, the largest productivity gain that can be realized by changing the assignment of operations to machines is $14\frac{2}{7}\%$. It is unknown whether this upper bound holds for $m \geq 5$.
- *Implementation issues*: The implementation issues discussed in Section 10 have open problems, including the following:
- The optimal use of local material handling devices remains an open question.
- The economic and performance trade-offs concerning schedules that revisit machines bear further study.

13. CONCLUDING REMARKS

We have presented a survey of recent work in robotic cell scheduling problems. We have discussed basic properties of robotic cells and detailed the tools most often used to analyze them. In so doing we have provided guidelines for determining the optimal policies for a variety of implementations. Furthermore, we have furnished a framework that future studies may employ to analyze new cells with different characteristics. This should also serve as a useful introduction to researchers and practitioners of the field. The material presented in this survey is the subject of a forthcoming book by Dawande et al. (to appear).

The field of robotic cell scheduling has grown rapidly over the past decade. As manufacturers have employed them in greater numbers and greater varieties, analysts have developed new models and new techniques to maximize these cells' economic utility. Competitive pressures will result in the development of more advanced cells and, hence, more sophisticated studies. Therefore, robotic cell scheduling should continue to attract the attention of a growing number of researchers.

ACKNOWLEDGMENT

The authors wish to thank the two anonymous referees and the associate editor, whose comments have significantly improved the content and the presentation of this paper.

APPENDIX: PROOF OF THEOREM 5

$T_a(\pi_D) = \max\{4m\delta + 2(m+1)\epsilon, \max p_i + 4(\delta + \epsilon)\}$. If $T_a(\pi_D) = \max p_i + 4(\delta + \epsilon)$, then it is optimal by Theorem 3. Assume $T_a(\pi_D) = 4m\delta + 2(m+1)\epsilon$. The set of pyramidal cycles contains an optimal cycle (Crama and van de Klundert, 1997a). Note that activity A_m is always considered to be an uphill activity, so π_D is the pyramidal cycle that corresponds to $U = \{m\}$.

Consider a general pyramidal cycle $\pi_p \neq \pi_D$, and let i, $1 \leq i \leq m-1$, be the smallest index of an uphill activity for cycle π_p. This implies that the form of cycle π_p is either

$$A_0 A_i A_{i+1} \ldots A_m \ldots A_{i-1} A_{i-2} \ldots A_1 \quad \text{or}$$

$$A_0 A_i \ldots A_m \ldots A_{i+1} A_{i-1} A_{i-2} \ldots A_1.$$

In either case, we can easily calculate lower bounds on the durations of the following nonoverlapping segments of the cycle:

1. from the start of activity A_i until the start of activity $A_{i+1} : \delta + 2\epsilon + p_{i+1}$
2. from the start of activity A_{i+1} until the start of activity $A_{i-1} : 4\delta + 2\epsilon$
3. from the start of activity A_{i-1} until the start of activity $A_i : \delta + 2\epsilon + p_i$

Thus, we have the following lower bound for the cycle time:

$$T(\pi_p) \geq p_i + p_{i+1} + 6\delta + 6\epsilon \geq 4m\delta + 2(m+1)\epsilon = T(\pi_D)$$

REFERENCES

Agnetis, A., "Scheduling no-wait robotic cells with two and three machines," *European Journal of Operational Research*, **123**, 303–314 (2000).

Agnetis, A. and D. Pacciarelli, "Part sequencing in three-machine no-wait robotic cells," *Operations Research Letters*, **27**, 185–192 (2000).

Akçalı, E., K. Nemoto, and R. Uzsoy, "Cycle-time improvements for photolithography process in semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, **14**, 48–56 (2001).

Aneja, Y. P. and H. Kamoun, "Scheduling of parts and robot activities in a two-machine robotic cell," *Computers and Operations Research*, **26**, 297–312 (1999).

Asfahl, C. R., *Robots and Manufacturing Automation*, John Wiley & Sons, New York, NY, 1985.

Baumann, W., R. Birner, J. Haensler, R. P. Hartmann, and A. B. Stevens, "Operating and idle times for cyclic multi-machine servicing," *The Industrial Robot*, 44–49 (1981).

Bedini, R., G. G. Lisini, and P. Sterpos, "Optimal programming of working cycles for industrial robots," *Journal of Mechanical Design. Transactions of the ASME*, **101**, 250–257 (1979).

Błażewicz, J., H. Eiselt, G. Finke, G. LaPorte, and J. Weglarz. "Scheduling tasks and vehicles in a flexible manufacturing system," *International Journal of Flexible Manufacturing Systems*, **4**, 5–16 (1991).

Błażewicz, J., S. P. Sethi, and C. Sriskandarajah, "Scheduling of robot moves and parts in a robotic cell," in K. E. Stecke and R. Suri (eds.), *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications* (Ann Arbor, MI), Elsevier Science Publishers, B.V., Amsterdam, The Netherlands, 1989, pp. 281–286.

Brauner, N. and G. Finke, "Final results on the one-cycle conjecture in robotic cells," Internal note, Laboratoire LEIBNIZ, Institut IMAG, Grenoble, France, 1997.

Brauner, N. and G. Finke, "Cyclic scheduling in a robotic flowshop," in *Proceedings IEPM'97, International Conference on Industrial Engineering and Production Management*, Lyon, France, 1997, vol.1, pp. 439–449.

Brauner, N. and G. Finke, "On a conjecture in robotic cells: new simplified proof for the three-machine case," *INFOR*, **37**(1), 20–36 (1999).

Brauner, N. and G. Finke, "Optimal moves of the material handling system in a robotic cell," *International Journal of Production Economics*, **74**, 269–277 (2001a).

Brauner, N. and G. Finke, "Cycles and permutations in robotic cells," *Mathematical and Computer Modeling*, **34**, 565–591 (2001b).

Brauner, N., G. Finke, and W. Kubiak, "Complexity of one-cycle robotic flow-shops," *Journal of Scheduling*, **6**, 355–371 (2003).

Che, A., C. Chu, and F. Chu, "Multicyclic hoist scheduling with constant processing times," *IEEE Transactions on Robotics and Automation*, **18**(1), 69–80 (2002).

Che, A., C. Chu, and E. Levner, "A polynomial algorithm for 2-degree cyclic robot scheduling," *European Journal of Operational Research*, **145**(1), 31–44 (2003).

Chen, H., C. Chu, and J. Proth, "Cyclic scheduling with time window constraints," *IEEE Transactions on Robotics and Automation*, **14**(1), 144–152 (1998).

Claybourne, B. H., "Scheduling robots in flexible manufacturing cells," *CME Automation*, **30**(5), 36–40 (1983).

Crama, Y., "Combinatorial optimization models for production scheduling in automated manufacturing systems," *European Journal of Operational Research*, **99**, 136–153 (1997).

Crama, Y., V. Kats, J. van de Klundert, and E. Levner, "Cyclic scheduling in robotic flowshops," *Annals of Operations Research: Mathematics of Industrial Systems*, **96**, 97–124 (2000).

Crama, Y. and J. van de Klundert, "Cyclic scheduling of identical parts in a robotic cell," *Operations Research*, **6**, 952–965 (1997a).

Crama, Y. and J. van de Klundert, "Robotic flowshop scheduling is strongly NP-complete," in W. K. Klein Haneveld et al. (eds.), *Ten Years LNMB*, CWI Tract, Amsterdam, 1997b, pp. 277–286.

Crama, Y. and J. van de Klundert, "Cyclic scheduling in 3-machine robotic flowshops," *Journal of Scheduling*, **2**, 35–54 (1999).

Dawande, M., N. Geismar, and S. Sethi, "Dominance of cyclic solutions and some open problems in scheduling bufferless robotic cells," to appear in *SIAM Review*, 2006a.

Dawande, M., N. Geismar, S. Sethi, and C. Sriskandarajah, *Throughput Optimization in Robotic Cells*, Springer, to appear.

Dawande, M., C. Sriskandarajah, and S. Sethi, "On throughput maximization in constant travel-time robotic cells," *Manufacturing and Service Operations Management*, **4**(4), 296–312 (2002).

Devedzic, V., "A knowledge-based system for the strategic control level of robots in flexible manufacturing cell," *International Journal of Flexible Manufacturing Systems*, **2**(4), 263–287 (1990).

Dixon, C. and S. D. Hill, "Work-cell cycle-time analysis in a flexible manufacturing system," in *Proceedings of the Pacific Conference in Manufacturing*, Sydney and Melbourne, Australia, 1990, vol. 1, pp.182–189.

Drobouchevitch, I. G., S. Sethi, and C. Sriskandarajah, "Scheduling dual gripper robotic cell: 1-unit cycles," to appear in *European Journal of Operational Research*, 2005.

Drobouchevitch, I. G., S. Sethi, J. Sidney, and C. Sriskandarajah, "A note on scheduling multiple parts in two-machine dual gripper robotic cell: Heuristic algorithm and performance guarantee," *International Journal of Operations and Quantitative Management*, **10**(4), 297–314 (2004).

Duenyas, I., J. W. Fowler, and L. W. Schruben, "Planning and scheduling in Japanese semiconductor manufacturing," *Journal of Manufacturing Systems*, 1994, pp. 323–333.

Finke, G., C. Gueguen, and N. Brauner, "Robotic cells with buffer space," in R. O'Connor and P. Magee (eds.), *ECCO IX Conference Proceedings*, Dublin City University, 1996.

Garey, M. R. and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.

Geismar, H. N., M. Dawande, and C. Sriskandarajah, "Robotic cells with parallel machines: Throughput maximization in constant travel-time cells," *Journal of Scheduling*, **7**, 375–395 (2004a).

Geismar, H. N., M. Dawande, and C. Sriskandarajah, "Approximation algorithms for *k*-unit cyclic solutions in robotic cells," *European Journal of Operational Research*, **162**, 291–309 (2005).

Geismar, H. N., M. Dawande, and C. Sriskandarajah, "Scheduling constant travel-time dual gripper robotic cells with parallel machines," *Working Paper*, School of Management, University of Texas at Dallas, 2004b.

Geismar, H. N., S. P. Sethi, J. B. Sidney, and C. Sriskandarajah, "A note on productivity gains in flexible robotic cells," *Working Paper*, School of Management, University of Texas at Dallas, 2004c.

Geismar, H. N., C. Sriskandarajah, and N. Ramanan, "Increasing throughput for robotic cells with parallel machines and multiple robots," *IEEE Transactions on Automation Science and Engineering*, **1**(1), 84–89 (2004d).

Gilmore, P. and R. Gomory, "Sequencing a one-state variable machine: A solvable case of the traveling salesman problem," *Operations Research*, **12**, 675–679 (1964).

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnoy Kan, "Optimization and approximation in deterministic sequencing and scheduling: A survey," *Annals of Discrete Mathematics*, **5**, 287–326 (1979).

Hall, N., "Operations research techniques for robotic systems," in S. Y. Nof (ed.), *Handbook of Industrial Robotics*, 2nd ed., John Wiley and Sons, 1999.

Hall, N. G., H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: Classification, two- and three-machine cells," *Operations Research*, **45**, 421–439 (1997).

Hall, N. G., H. Kamoun, and C. Sriskandarajah, "Scheduling in robotic cells: Complexity and steady state analysis," *European Journal of Operational Research*, **109**, 43–63 (1998).

Hall, N. G. and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Operations Research*, **44**(3), 510–525 (1996).

Hall, N. G., C. Potts, and C. Sriskandarajah, "Parallel machine scheduling with a common server," *Discrete Applied Mathematics*, **102**, 223–243 (2000).

Herrmann, J., N. Chandrasekaran, B. Conaghan, M. Nguyen, G. Rubloff, and R. Shi, "Evaluating the impact of process changes on cluster tool performance," *IEEE Transactions on Semiconductor Manufacturing*, **13**, 181–192 (2000).

Ioachim, I. and F. Soumis, "Schedule efficiency in a robotic production cell," *The International Journal of Flexible Manufacturing Systems*, **7**, 5–26 (1995).

Kamoun, H., "Scheduling in repetitive manufacturing systems: Complexity, heuristic algorithms and system design," Ph.D. Thesis, Graduate Department of Industrial Engineering, University of Toronto, 1994.

Kamoun, H., N. G. Hall, and C. Sriskandarajah, "Scheduling in robotic cells: Heuristics and cell design," *Operations Research*, **47**, 821–835 (1999).

Kats, V., "An exact optimal cyclic scheduling algorithm for multioperator service of a production line, part 2," *Automation and Remote Control*, **42**(4), 538–543 (1982).

Kats, V. and E. Levner, "Minimizing the number of robots to meet a given schedule," *Annals of Operations Research*, **69**, 209–226 (1997).

Kats, V. and E. Levner, "Polynomial algorithms for cyclic scheduling of tasks on parallel processors," in *Proceedings of the 16th IASTED International Conference on Applied Informatics*, Garmisch, Germany, 1998, pp. 302–304.

Kats, V. and E. Levner, "Cycle scheduling in a robotic production line," *Journal of Scheduling*, **5**, 23–41 (2002).

Kats, V., E. Levner, and L. Meyzin, "Multiple-part cyclic hoist scheduling using a sieve method," *IEEE Transactions on Robotics and Automation*, **15**(4), 704–713 (1999).

Kondoleon, A. S., "Cycle time analysis of robot assembly systems," in *Proceedings of the Ninth Symposium on Industrial Robots*, 1979, pp. 575–587.

Kumar, S., N. Ramanan, and C. Sriskandarajah, “Minimizing cycle time in large robotic cells,” *IIE Transactions*, **37**(2), 123–136 (2005).

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (eds.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, Chichester, 1985.

Lei, L. and T. J. Wang, “Determining optimal cyclic hoist schedules in a single-hoist electroplating line,” *IIE Transactions*, **26**(2), 25–33 (1994).

Levner, E., V. Kats, and V. Levit, “An improved algorithm for cyclic flowshop scheduling in a robotic cell,” *European Journal of Operational Research*, **97**, 500–508 (1997).

Little, J. D. C., “A proof for the queuing formula: $L = \lambda W$,” *Operations Research*, **9**(3), 383–387 (1961).

Logendran, R. and C. Sriskandarajah, “Sequencing of robot activities and parts in two-machine robotic cells,” *International Journal of Production Research*, **34**(12), 3447–3463 (1996).

Maimon, O. Z. and S. Y. Nof, “Coordination of robots sharing assembly tasks,” *Journal of Dynamic Systems Measurement and Control. Transactions of the ASME*, **107**(4), 299–307 (1985).

Mangione, F., N. Brauner, and B. Penz, “Optimal cycles for the robotic balanced no-wait flowshop,” in *Proceedings IEPM’03, International Conference of Industrial Engineering and Production Management*, Porto, Portugal, May 2003.

Medeiros, D. J., E. F. Watson, J. S. Carson, and M. S. Manivannan (eds.), “Operational modeling & simulation in semiconductor manufacturing,” in *Proceedings of the 1998 Winter Simulation Conference*.

Miller, R. K., *Robots in Industry: Applications for the Electronics Industry*, SEAI Institute, New York, 1984.

Nof, S. Y. and D. Hannah, “Operational characteristics of multi-robot systems with cooperation,” *International Journal of Production Research*, **27**(3), 477–492 (1989).

Perkinson, T., P. McLarty, R. Gyurcsik, and R. Cavin, “Single-wafer cluster tool performance: An analysis of throughput,” *IEEE Transactions on Semiconductor Manufacturing*, **7**, 369–373 (1994).

Perkinson, T., R. Gyurcsik, and P. McLarty, “Single-wafer cluster tool performance: An analysis of the effects of redundant chambers and revisitation sequences on throughput,” *IEEE Transactions on Semiconductor Manufacturing*, **9**, 384–400 (1996).

Pinedo, M., *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1995.

Ramanan, N., personal communication, 2002.

Rudge, D. A., “The automation of solution phase synthetic chemistry using XP zymate laboratory robotic systems,” *Laboratory Automation and Information Management*, **33**, 81–86 (1997).

Sethi, S. P., C. Sriskandarajah, G. Sorger, J. Błażewicz, and W. Kubiak, “Sequencing of parts and robot moves in a robotic cell,” *Int. J. Flexible Manufacturing Systems*, **4**, 331–358 (1992).

Sethi, S. P., J. Sidney, and C. Sriskandarajah, “Scheduling in dual gripper robotic cells for productivity gains,” *IEEE Transactions on Robotics and Automation*, **17**, 324–341 (2001).

Sriskandarajah, C., I. G. Drobouchevitch, S. Sethi, and R. Chandrasekaran, “Scheduling multiple parts in a robotic cell served by a dual gripper robot,” *Operations Research*, **52**, 65–82 (2004).

Sriskandarajah, C., N. G. Hall, and H. Kamoun, “Scheduling large robotic cells without buffers,” *Annals of Operations Research*, **76**, 287–321 (1998).

Su, Q. and F. Chen, “Optimal sequencing of double-gripper gantry robot moves in tightly-coupled serial production systems,” *IEEE Transactions on Robotics and Automation*, **12**, 22–30 (1996).

Suri, R., “Quantitative techniques for robotic systems analysis,” in S. Y. Nof (ed.), *Handbook of Industrial Robotics*, John Wiley and Sons, 1985, , Vol. I.

Van de Klundert, J., *Scheduling Problems in Automated Manufacturing*, Faculty of Economics and Business Administration, University of Limburg, Maastricht, The Netherlands, Dissertation no. 96-35, 1996.

Venkatesh, S., R. Davenport, P. Foxhoven, and J. Nulman, “A steady-state throughput analysis of cluster tools: Dual-blade versus single-blade robots,” *IEEE Transactions on Semiconductor Manufacturing*, **10**, 418–424 (1997).

Wilhelm, W. E., “Complexity of sequencing tasks in assembly cells attended by one or two robots,” *Naval Research Logistics*, **34**, 721–738 (1987).

Wittrock, R. J., “Scheduling algorithms for flexible flow lines,” *IBM Journal of Research and Development*, **29**, 401–412 (1985).

Wood, S., “Simple performance models for integrated processing tools,” *IEEE Transactions on Semiconductor Manufacturing*, **9**, 320–328 (1996).

WEB SITES REFERENCES

Applied Robotics, Inc., http://www.arobotics.com/product.html, 2001.
Asia Pacific Metal Working, http://www.equipment-news.com/magazines/2000/sep2000/art05.htm, 2000.
Glass on Web, http://www.glassonweb.com/articles/article/13/, 2002.
Manufacturing Talk, http://www.manufacturingtalk.com/news/bab/bab101.txt, 2001.
Manufacturing Talk, http://www.manufacturingtalk.com/news/guy/guy123.txt, 2001.
Products Finishing Magazine, http://www.pfonline.com/articles/080103.html, 2001.
Rapid Development Services, Inc., http://roboticintegrator.com/, 2001.
N. Remich, "The Robotic Ballet," *Appliance Manufacturer*, http://www.ammagazine.com, 2000.
RoboDesign International, Inc., http://www.robodesign.com/roboarrayer_options.htm, 2002.
Robot Workspace Technologies, Inc., http://www.rwt.com/RWT_Content_Files/articles/RWT_ASept99ME.html, 1999.
Sawyer and Smith Corporation, http://www.buffingandpolishing.com/automaticproducts.htm, 2001.
Smart Media Group Ltd, http://www.robotics-technology.com/articlearchive/2001/112001.shtml, 2001.
Vulcan Publications, Inc., http://www.ndx.com/article.asp?article_id=270 & channel_id=6, 2001.
Zmation, Inc., http://www.zmation.com/products.htm, 2002.