SCHEDULING DUAL GRIPPER ROBOTIC CELL:

ONE-UNIT CYCLES

Inna G. Drobouchevitch†

Suresh P. Sethi *

Chelliah Sriskandarajah*

* The University of Texas at Dallas, School of Management, Richardson, Texas 75083, U.S.A

November 2000

Revised November 2003
Abstract

We consider the scheduling problem of cyclic production in a bufferless dual-gripper robot cell processing a family of identical parts. The objective is to find an optimal sequence of robot moves so as to maximize the long-run average throughput rate of the cell. While there has been a considerable amount of research dealing with single-gripper robot cells, there are only a few papers devoted to scheduling in dual-gripper robotic cells. From the practical point of view, the use of a dual gripper offers the attractive prospect of an increase in the cell productivity. At the same time, the increase in the combinatorial possibilities associated with a dual-gripper robot severely complicates its theoretical analysis. The purpose of this paper is to extend the existing conceptual framework to the dual-gripper situation, and to provide some insight into the problem.

We provide a notational and modelling framework for cyclic production in a dual-gripper robotic cell. Focusing on the so-called active cycles, we discuss the issues of feasibility and explore the combinatorial aspects of the problem. The main attention is on 1–unit cycles, i.e., those that restore the cell to the same initial state after the production of each unit. For an $m$–machine robotic cell served by a dual-gripper robot, we describe a complete family of 1–unit cycles, and derive an analytical formula to estimate their total number for a given $m$. In the case when the gripper switching time is sufficiently small, we identify an optimal 1–unit cycle. This special case is of particular interest as it reflects the most frequently encountered situation in real-life robotic systems. Finally, we establish the connection between a dual-gripper cell and a single-gripper cell with machine output buffers of one-unit capacity and compare the cell productivity for these two models.

Key words: Scheduling, robotic cells, cyclic production, 1–unit cycles.
1 Introduction

A robotic cell is an automated conglomerate of robot material handler(s), machines with specifically designed part processing capability and flexibility, and a dynamic (computerized) control logic device coordinating all movements. We consider the problem of scheduling operations in a bufferless dual-gripper robotic cell producing single part-types. A robotic cell contains a number of machines served by a robot [2, 16, 19]. The movable robot arm comprises two grippers, either of them can be in possession of at most one part at a time. We assume that the order in which parts go through the machines is the same for all parts. This allows us to treat the cell as a flow shop. The objective is to maximize the throughput rate, or equivalently, to minimize the steady state cycle time to produce one part. In the case of identical parts production, the problem is reduced to finding an optimal sequence of robot moves to be executed repeatedly in a cyclic manner. A four-machine robot cell with a dual-gripper robot is illustrated in Figure 1.

A cyclic schedule is one in which the same sequence of states is repeated over and over again (with the obvious proviso that the specific parts in the system are changing). A cycle in such a schedule begins at any given state and ends when that state is encountered next. In each execution of a cyclic schedule, one or more parts will be completed. If \( q \) parts are produced in a cycle, we call the latter a \( q \)-unit cycle. In this case, the per unit cycle time equals to \( \frac{1}{q} \) times the total cycle time required to produce \( q \) units. In this paper, we primarily focus our analysis on 1–unit cycles, and we shall be interested in the steady state operations of the system under various cyclic scheduling options. One-unit cycles are attractive from a practical point of view because of their conceptual simplicity and their ease of implementation. From the theoretical point of view, systematic characterization of cyclic solutions is essential in a scheduler’s quest for the optimality. For dual-gripper robot cells, defining 1–unit cycles is a natural starting point in this direction. In this paper, we propose a conceptual framework to develop and describe 1–unit cycles.

In a cell served by a single-gripper robot, neither the machines nor the robot can be in possession of more than one part at any time. In contrast, a dual-gripper robot is able to handle at most two parts at a time. Such a possibility leads to a dramatic rise in the number of feasible robot moves strategies. Consequently, a dual-gripper robot problem is much more complicated than a single-gripper one. Owing to its inherent complexity, the cell with a dual-gripper robot, even if it is much more productive, evidently presents a challenge in conducting an analytical investigation. We are aware of only a few published papers on the subject [25, 27, 28]. Su and Chen [28] conduct the study on a two-machine system producing identical parts. Essentially, their study concerns an examination of only five
out of 46 possible 1–unit cycles (See Table 1). For \( m \)-machine cells, Sethi et al. [25] have examined the cycle time advantage (or throughput advantage) of using a dual gripper in place of a single gripper robot. They show that an \( m \)-machine dual-gripper cell may provide at most a 100% increase in productivity over a single-gripper. They also provide a computational study on the productivity gains for realistic problem instances.

Our motivation is to extend the existing single-gripper framework to the dual-gripper robot cell problem. We develop an analytical approach to address the problem and provide some insight into the problem solving. The main attention is on 1–unit cycles. Focusing on the so-called active cycles, we discuss the issues of feasibility and explore the combinatorial aspect of the problem. We derive the number of all active 1–unit cycles. We also develop an algorithmic framework to construct all 1–unit cycles.

The remainder of the paper is organized as follows. We provide a brief literature review in the next section. In Section 3, we formulate our robotic cell scheduling problem along with the required notation. Section 4 presents some general observations and notions for dual-gripper robot cycles. In Section 5, we provide a method to describe 1–unit cycles and explore their basic properties. In Section 6, we focus on the problem of generating all possible 1–unit active cycles for an \( m \)-machine cell. Section 7 is devoted to a special case of the problem for which an optimal 1–unit cycle can be found efficiently. In Section 8, we establish the connection between a dual gripper cell and a single gripper cell having one unit output buffers at each machine, and compare their productivity. Section 9 concludes the paper with some remarks.
2 Literature Review

Robotic cells utilizing a robot with a single gripper as its material handling device have received considerable amount of attention in the literature since the pioneering work of Sethi et al. [24]. It has been shown that there are \( m! \) 1–unit cycles in an \( m \) machine single-gripper robot cell [24]. Crama and van de Klundert [10] describe a polynomial-time dynamic programming algorithm for finding the best 1–unit cycle in an \( m \)-machine cell producing a single part-type.

In [24], Sethi et al. put forward the following conjecture: *In bufferless single-gripper robot cells producing single part-type and having identical robot travel times between adjacent machines and identical load/unload times, a 1–unit cycle provides the minimum per unit cycle time in the class of all solutions, cyclic or otherwise.* The conjecture is now resolved. In the two-machine case, Sethi et al. [24] prove that a 1–unit cycle solution is optimal over the class of all solutions, cyclic or otherwise. In the three-machine case, Crama and van de Klundert [11] and Brauner and Finke [4] show that the best 1–unit cycle is optimal among the class of all cyclic solutions. For four-machine cells, Brauner and Finke [3, 5, 6] have shown that there exist problem instances for which 1–unit cycles can be dominated by 2–unit cycles in case of nonidentical robot travel times, and by 3–unit cycles - in an equidistant case (identical robot travel times between adjacent machines).

Scheduling problems encountered in single-gripper robotic cells producing multiple parts have been studied extensively in the literature [17, 18, 20, 23, 26]. Hall, Kamoun and Sriskandarajah [17, 18] consider a three-machine cell producing multiple part-types, and prove that in two out of the six possible robot move sequences, the recognition version of the part sequencing problem is unary NP-complete, while the other four cycles are solvable in polynomial time. Kamoun, Hall and Sriskandarajah [20] describe and test an efficient heuristic procedure for the combined robot move and part sequencing problem in robotic cells. They also consider robotic cells designs for efficient performance by grouping machines into cells, identifying good part sequences, and providing appropriate size buffers between cells, in a larger manufacturing system. Sriskandarajah, Hall and Kamoun [26] study \( m \)-machine robotic cells, for \( m \geq 2 \). They show that the part sequencing problems associated with \( 2m - 2 \) of the \( m! \) feasible robot move sequences are solvable in polynomial time. The remaining cycles have associated part sequencing problems which are unary NP-hard, and some of these can be modeled as traveling salesman problems.

For detailed reviews of the relevant literature, the reader is referred to Crama [8], Crama et al. [9], Sethi
et al. [24], Sriskandarajah, Hall and Kamoun [26], Dawande et al. [13].

Due to the dual gripper flexibility, the structure of the problem changes dramatically and a new framework is needed to be developed. A very few studies deal with dual-gripper robot cells (Sethi et al. [25], Su and Chen [28]). These papers address problems in two-machine cells to minimize the cycle time in the context of the single part-type production. In this paper, we study the problem of scheduling operations in a dual-gripper bufferless robotic cell consisting of \( m \) machines producing single part type.

In conclusion, we briefly mention other robotic cell models/applications that are related to the model considered in this paper. One of the robotic systems studied in the scheduling literature is a single hoist electroplating line (see for example, Lei and Wang [21]). In this problem, a single-hoist electroplating line consisting of \( m \) processing tanks (i.e., machines) arranged in series is used to apply chemical or plating treatments on printed circuit boards (i.e., parts). The hoist (i.e., robot) handles the inter-tank movements of the parts. A tank can process one part at a time, and a job cannot pause between the tanks, due to deterioration of the part while exposed to the atmosphere. Each part must be immersed in tank \( M_i \) for some time \( p_i \), \( u_i \leq p_i \leq v_i \), for given bounds \( u_i \) and \( v_i \), \( i = 1, \ldots, m \). When \( v_i = u_i \) for each \( i \), the problem reduces to the stranded no-wait problem (see for example, Agnetis [1], Levner et al. [22]). In this case, a job must be processed for time \( p_i = u_i \) exactly, and then immediately removed and loaded onto the next machine. When \( v_i = \infty \) for each \( i \), then we have a bufferless environment being considered in this paper. Here, the processing time of part \( i \) is always \( p_i = u_i \), but after its processing a job may remain on the machine for any amount of time before the robot unloads it.

3 Preliminaries

We start with a formal description of the system under investigation. The system consists of \( m \) machines \( (M_1, M_2, \ldots, M_m) \) without any buffer capacity and an I/O-station (also called \( M_0 \) or \( M_{m+1} \) when referred to as a machine), all served by a dual-gripper robot. The robot performs transportation of parts and machine load/unload operations. The robot arm consists of two grippers, each of which can be in possession of at most one part. The I/O station consists of an Input device, from which parts are introduced into the cell, and an Output device, onto which the parts are dropped upon completion of their processing on the machines. Each part is processed first on machine \( M_1 \), then on machine \( M_2 \), and so on, until it is processed on the last machine \( M_m \). Thus, the part processing route is \( I/O(Input), M_1, M_2, \ldots, M_m, I/O(Output) \), the same for all parts. At any time a machine can process
at most one part. The processing time $p_i$ of a part on machine $M_i$, $i = 1, 2, \ldots, m$, is known in advance. The durations of loading/unloading operations and the robot travel times between any pair of adjacent machines are given:

$\delta$ : the time taken by a rotational robot movement when traveling between two consecutive machines $M_{j-1}$ and $M_j$, $1 \leq j \leq m + 1$ (here, both $M_0$ and $M_{m+1}$ denote the I/O–station: $M_0$ stands for the Input buffer while $M_{m+1}$ refers to the Output buffer of I/O). Furthermore, the travel durations are additive for non-consecutive machines, i.e., the trip time of the robot from $M_i$ to $M_j$ is $\delta \times \min \{|i - j|, m + 1 - |i - j|\}$. The robot travel time between locations $x$ and $y$ is denoted by $\ell(x, y)$. For example, in a seven-machine robot cell, $\ell(I/O, M_1) = \delta$, $\ell(M_1, M_3) = 2\delta$, $\ell(M_1, M_7) = 2\delta$, $\ell(M_3, I/O) = 3\delta$, $\ell(M_5, M_2) = 3\delta$, etc. We remark that such cells (i.e., with additive travel times for non-consecutive machines) are called additive travel time cells.

$\epsilon$ : the time taken by the robot to pick up/drop off a part at I/O; also the time taken by the robot to perform a load /unload operation at any machine.

$\theta$: the grippers switching time which is the time required for the dual-gripper robot to reposition itself immediately after one gripper has unloaded a machine, so that its second gripper is positioned to load the same machine.

$\theta_t$: the gripper switching time, which is the time required by the robot to reposition its grippers while traveling from one machine to another, i.e., when there are two immediately successive operations that are executed on different machines and requiring different grippers. In such a situation, the robot performs the grippers switch while traveling between the machines. Thereby, if there are two successive operations executed on machines $M_i$ and $M_j$ that require different grippers, the time taken by the robot, which has just finished serving $M_i$, to reposition itself for its next operation at $M_j$ is $\max\{\theta_t, \ell(M_i, M_j)\}$.

The objective is to find a cyclic schedule which maximizes the long-run average production rate of a single part-type to be manufactured in the cell.

For the system under consideration, we define the notion of state. The state of a robotic cell at an instant of time may be thought of as a snapshot of the system taken at this particular point in time. Such a snapshot provides the full information on the system. In particular, it gives the information on the state of each machine: vacant or loaded with a part, and if loaded, the amount of time the part has been residing on this machine; the precise location of a robot as well as the state of robot grippers: empty or occupied with a part, and if occupied, a machine onto which the part is going.
As an example of a state of the robotic cell with three machines consider the following snapshot. A robot is positioned half-way between machines $M_1$ and $M_2$; one of the robot grippers is occupied with a part intended for machine $M_2$; the other gripper is empty; machines $M_1$ and $M_2$ are loaded, machine $M_3$ is empty; the part on $M_1$ is undergoing processing, the remaining processing time is 5 min; the part on $M_2$ has just finished its processing.

A $q$–unit cycle (or $q$–cycle) $C$ is defined as a sequence of robot moves where exactly $q$ parts enter the system at Input, $q$ parts leave the system at Output, and the complete execution of the cycle restores the system to the same initial state. This allows the $q$–unit cycle to be repeated indefinitely. The aim is to maximize the throughput rate $q/T(C)$, or equivalently, to minimize the cycle length (or per unit cycle time) $T(C)/q$, where $T(C)$ is the cycle time.

We adapt the standard classification scheme (Graham et al. [15]) to denote the scheduling problems arising in robotic cells. In particular, $RF^2_m |q = 1| C_t$ denotes the minimization of the per unit cycle time ($C_t$) in an $m$–machine dual-gripper robotic flow shop ($RF^2_m$) producing identical parts using 1–unit cycles ($q = 1$).

4 Active cycles (schedules) and $S$–sequences

In this Section, we introduce some basic concepts and notions to describe the cyclic scheduling of operations in a dual-gripper robot cell.

A robot in a robotic cell performs three kinds of operations: loading parts onto machines, unloading parts from machines, and transportation of parts from one machine to another. We describe a robot schedule in terms of a “load/unload machine” activities. Obviously, such a sequence would uniquely define the travel of the robot between machines (i.e., from/to which machine the robot travels, and when.)

**Definition 1** A sequence of robot activities “unload a machine $M_i$”, $i = 0, 1, \ldots, m$, and “load a machine $M_i$”, $i = 1, 2, \ldots, m + 1$, is called **feasible** if

(i) the robot never has to unload any empty machine;

(ii) the robot never has to load any loaded machine;

(iii) the robot never carries more than two parts, or equivalently, at least one of the robot’s grippers has to be empty whenever the robot wants to pick up a part from a machine.
Definition 2 A \( q \)-unit robot cycle is defined as a sequence of robot activities accomplishing exactly \( q \) parts to enter the system at Input device of I/O−buffer, exactly \( q \) parts to leave the system at Output device of I/O−buffer, and providing the restoration of the system to the same state upon complete execution of the robot activities sequence.

Property 1 In a \( q \)-unit robot cycle executed in an \( m \)-machine cell, each of the operations “unload machine \( M_i \)”, \( i = 0, 1, \ldots, m \), and “load machine \( M_i \)”, \( i = 1, 2, \ldots, m+1 \), is performed exactly \( q \) times.

Proof. By Definitions 1 and 2. \( \square \)

We will focus on so-called active cycles.

Definition 3 A robot cycle is called non-active if the robot has a part (i.e., regardless of the gripper holding the part), intended for any specific machine, in its possession continuously for an amount of time equal to a cycle time. A robot cycle is called active if it is not non-active.

Example 1: non-active \( 1 \)−unit cycle for \( m = 2 \)

Consider the following robot cycle described by the sequence of robot activities and an initial state of robot grippers. A robot move sequence: pick up a part from \( I \)−buffer; load a part on machine \( M_1 \); unload a part from machine \( M_1 \); load a part on machine \( M_2 \); unload a part from machine \( M_2 \); drop a part onto \( O \)−buffer. Initial state: the robot is positioned at \( I/O \) hopper ready to pick up a part; one of its grippers holds a part to be loaded on machine \( M_2 \) and its other gripper is empty. Machines \( M_1 \) and \( M_2 \) are empty. What happens in this cycle is that the robot comes to \( M_2 \) with each of its grippers holding a part intended for that machine. It is easy to see that, no matter which part is chosen to be loaded onto machine \( M_2 \), the other part would occupy the gripper continuously until the next chance for it to be unloaded from the gripper, and it will not happen until the point when the robot comes to load machine \( M_2 \) again, that is, not sooner than in time equal to a cycle time. And so, we have a non-active cycle.

The definition of active/non-active cycles straightforwardly implies the following observations.

Remark 1 For an active cycle, the transportation of a part from machine \( M_i \) to machine \( M_{i+1} \) always takes less than the cycle time, for all \( i \in \{0, 1, \ldots, m\} \).
**Remark 2** Essentially, by restricting our attention to active cycles, we exclude from the consideration the cycles, which are executable by a single-gripper robot (one gripper only utilization is sufficient), but yet executed by a dual-gripper robot with both grippers in use (both grippers hold parts during some period of the cycle).

**Remark 3** For an active cycle, if a gripper does not perform any operation during a cycle execution, it is presumed to be empty.

**Lemma 1** In a set of all $q$–unit cycles, there exists an optimal $q$–unit cycle which is an active cycle, for any given $q$.

Proof. The proof follows straightforwardly from the above Remarks. Indeed, a robot moves sequence of any non-active optimal cycle can be executed by a single-gripper robot. Thus, any non-active optimal cycle can be substituted by its active counterpart defined by the same sequence of robot moves and executed by the robot with one of its grippers being empty continuously during the cycle execution. Trivially, the cycle length will remain the same. □

Furthermore, from the above Remarks we have that, for an active cycle, the sequence of robot activities uniquely determines all operations in a cell, as well as the states of machines and robot grippers (empty or occupied), at any moment of time. This yields the following

**Property 2** A feasible sequence of robot activities “unload a machine $M_i$”, $i = 0, 1, \ldots, m$, and “load a machine $M_i$”, $i = 1, 2, \ldots, m + 1$, to represent a robot cycle uniquely defines an active robot cycle.

For an $m$–machine robotic cell, we will describe a sequence of robot activities “unload a machine” and “load a machine” in terms of $x^-$ and $x^+$ notion, $x \in \{0, 1, \ldots, m\}$, where $x^-$ reads “unload machine $M_x$”, and $x^+$ stands for “load machine $M_{x+1}$”. By virtue of Property 2, any $q$–unit robot cycle could be then described by a sequence $S = (s_1, s_2, \ldots, s_{2q(m+1)})$, where $s_j \in \{x^-, x^+ | x \in \{0, 1, \ldots, m\}\}$, $j = 1, 2, \ldots, 2q(m+1)$. We also may straightforwardly employ Definition 1 for defining feasible $S$–sequences.

**Example 2:** $S$–sequence to represent an active robot cycle

- The example is to illustrate the introduced notion of $S$–sequences. Given a three machine robotic cell, let

$$S = (0^-, 0^+, 2^-, 1^+, 3^-, 2^+, 1^-, 3^+)$$
The sequence \( S \) translates into the sequence of robot activities as follows: pick up a part from the Input (I/O−hopper); load machine \( M_1 \), unload machine \( M_2 \), load machine \( M_2 \), unload machine \( M_3 \), load machine \( M_3 \), unload machine \( M_1 \), drop a part onto Output (I/O−hopper). So, at the outset of the cycle execution, the robot is at the I/O−hopper ready to pick up a part. The initial state of robot grippers is uniquely predetermined by the sequence of robot moves and as follows: one gripper is empty, and the other is occupied by the part, which goes to machine \( M_2 \).

From now on, we will concentrate exclusively on 1−unit cycles.

5 1−unit active cycles and their representation by \( S \)−sequences

Following the scheme introduced in the previous section, we use \( S \)−sequences to represent 1−unit cycles. Namely, for an \( m \)−machine robotic cell served by a dual-gripper robot, we deal with \( S \)−sequences of the form

\[
S = (s_1, s_2, \ldots, s_{2(m+1)}),
\]

where

\[
s_j \in \{x^-, x^+|x \in \{0,1,\ldots,m\}\}, \ j = 1,2,\ldots,2(m+1);
\]

\[
s_j \neq s_i, \ \text{for} \ i \neq j, \ i,j \in \{1,2,\ldots,2(m+1)\}.
\]

Furthermore, given a sequence \( S \) as defined above, we also use the notation \( p_S(*) \) to denote the position of a particular \( s_j = * \) in \( S \), namely, \( p_S(s_j) = j \). For example, for \( S = (0^-, 3^+, 2^-, 2^+, 1^-, 0^+, 3^-, 1^+) \), we would have \( p_S(0^-) = 1, p_S(3^+) = 2, p_S(2^-) = 3 \), and so on. Note that in the single gripper case, \( p_S(x^+) = p_S(x^-) + 1 \) in any feasible sequence.

Our objective is to establish a one-to-one correspondence between the set of all active cycles and the set of (feasible) \( S \)−sequences, which “notationally” represent these cycles.

**Property 3** A sequence \( S \) of robot activities “unload a machine”/“load a machine” to define a 1−unit robot cycle is feasible iff at any moment of time, the cycle does not require from the robot to carry more than two parts.
Proof. Follows from Definition 1, Property 1, and a simple observation that conditions (i) and (ii) of Definition 1 are always satisfied for a 1–unit cycle when the latter is written by a sequence of “unload a machine”/“load a machine”-operations. □

Given a sequence \( S = (s_1, s_2, \ldots, s_{2(m+1)}) \) defined by (1)-(3) and \( k \in \{0, 1, \ldots, m\} \), let

\[
S_k = (s_l = k^-, s_{l+1}, \ldots, s_{2(m+1)}, s_1, s_2, \ldots, s_{l-1})
\]  

be a sequence constructed from \( S \). That is a sequence \( S_k, k \in \{0, 1, \ldots, m\} \), is obtained from the sequence \( S \) by treating the latter in a cyclic manner and rewriting it to begin with activity \( k^- \), i.e., the activity “unload \( M_k \)”.

The following lemma reformulates Property 3 and gives explicit conditions to guarantee that the robot is never forced to carry more than two parts.

**Lemma 2** A sequence \( S \) defined by (1)-(3) is feasible iff each of sequences \( S_k, k \in \{0, 1, \ldots, m\} \), constructed from \( S \) as specified by (4), satisfies the following requirement:

for any \( r \in \{0, 1, \ldots, m\} \setminus \{k\} \) such that

\[
p_{S_k}(r^-) < p_{S_k}(k^+),
\]

one must have either

\[
p_{S_k}(r^-) + 1 = p_{S_k}(k^+),
\]

or

\[
p_{S_k}(r^-) + 1 = p_{S_k}(r^+).
\]

Proof. The proof follows in a straightforward manner from Property 3. Indeed, to insure the feasibility of a sequence of robot activities we must guarantee the following: if the robot with one of its grippers occupied is scheduled to unload a machine, and so - to pick up another part, the next operation must be “unload the robot”. In other words, for any two “unload a machine” operations which result in both grippers of the robot being occupied, say for “unload \( M_k \)” being followed, at some stage, by “unload \( M_r \)”, \( k, r \in \{0, 1, \ldots, m\} \), (with “load \( M_{k+1} \)” being not executed in meantime!), the only feasible robot activity to be executed next is “to empty one of the robot’s grippers”, which means that either “load \( M_{k+1} \)” or “load \( M_{r+1} \)” must be executed. This requirement, if written in terms of \( S \)-sequences notation,
and checked for all pairs of any two successive (in cyclic sense) “unload a machine” operations, is exactly what the lemma claims. □

**Lemma 3** A sequence $S = (s_1, s_2, \ldots, s_{2(m+1)})$ given by (1)-(3) is feasible iff all the following five conditions (i)-(v) hold:

(i) for any pair of $x$ and $y$, $x, y \in \{0, 1, \ldots, m\}$, such that

$$p_S(x^-) < p_S(y^-) < \min\{p_S(x^+), p_S(y^+)\},$$

one must have

$$p_S(y^-) + 1 = \min\{p_S(x^+), p_S(y^+)\}.$$ 

(ii) for any pair of $x$ and $y$, $x, y \in \{0, 1, \ldots, m\}$, such that

$$p_S(x^-) < \min\{p_S(x^+), p_S(y^+)\} < \max\{p_S(x^-), p_S(y^-)\} < p_S(y^-),$$

one must have

$$p_S(x^-) + 1 = \min\{p_S(x^+), p_S(y^+)\}.$$ 

(iii) for any pair of $x$ and $y$, $x, y \in \{0, 1, \ldots, m\}$, such that

$$p_S(x^+) < \min\{p_S(x^-), p_S(y^-)\} < \max\{p_S(x^-), p_S(y^-)\} < p_S(y^+),$$

one must have

$$\max\{p_S(x^-), p_S(y^-)\} + 1 = p_S(y^+).$$ 

(iv) for any pair of $x$ and $y$, $x, y \in \{0, 1, \ldots, m\}$, such that

$$p_S(x^-) < p_S(y^+) < p_S(y^-) < p_S(x^+),$$

one must have

$$p_S(x^-) + 1 = p_S(y^+) \quad (8)$$

and

$$p_S(y^-) + 1 = p_S(x^+). \quad (9)$$
(v) for any pair of \( x \) and \( y \), \( x, y \in \{0, 1, \ldots, m\} \), such that \( p_S(x^+) < p_S(x^-) \) and \( p_S(y^+) < p_S(y^-) \), one must have

\[
\min\{p_S(x^+), p_S(y^+)\} = 1 \quad (10)
\]

\[
\max\{p_S(x^-), p_S(y^-)\} = 2(m + 1) \quad (11)
\]

Furthermore, if

\[
\min\{p_S(x^-), p_S(y^-)\} < \max\{p_S(x^+), p_S(y^+)\} \quad (12)
\]

then one must also have

\[
\min\{p_S(x^-), p_S(y^-)\} + 1 = \max\{p_S(x^+), p_S(y^+)\}. \quad (13)
\]

Proof. The proof is given in Appendix A. □

Comment 1 We may show that each of conditions (i)-(v) is vital for ensuring the feasibility of an \( S \)-sequence. Below, we give the examples of \( S \)-sequences, which violate exactly one of (i)-(v) conditions, all possible situations of the violation of each condition are portrayed (in all examples, the violation of the corresponding condition is always for \( x = 0 \) and \( y = 1 \)).

<table>
<thead>
<tr>
<th>condition, which is violated</th>
<th>sequence ( S )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)</td>
<td>( 0^-, 1^-, 2^-, 2^+, 0^+, 1^+ )</td>
</tr>
<tr>
<td></td>
<td>( 0^-, 1^-, 2^-, 2^+, 1^+, 0^+ )</td>
</tr>
<tr>
<td>(ii)</td>
<td>( 0^-, 2^-, 0^+, 2^+, 1^+, 1^+ )</td>
</tr>
<tr>
<td></td>
<td>( 2^-, 0^-, 2^+, 1^+, 0^+, 1^+ )</td>
</tr>
<tr>
<td>(iii)</td>
<td>( 0^+, 0^-, 1^-, 2^-, 2^+, 1^+ )</td>
</tr>
<tr>
<td></td>
<td>( 0^+, 1^-, 0^-, 2^-, 2^+, 1^+ )</td>
</tr>
<tr>
<td>(iv)</td>
<td>( 0^-, 2^-, 2^+, 1^+, 1^+, 0^+ )</td>
</tr>
<tr>
<td></td>
<td>( 0^-, 1^+, 1^-, 2^-, 2^+, 0^+ )</td>
</tr>
<tr>
<td>(v)</td>
<td>( 2^-, 2^+, 0^+ , 1^+, 1^-, 0^- )</td>
</tr>
<tr>
<td></td>
<td>( 2^-, 2^+, 0^+, 1^+, 0^-, 1^- )</td>
</tr>
<tr>
<td></td>
<td>( 0^+, 1^+, 1^-, 0^-, 2^- , 2^+ )</td>
</tr>
<tr>
<td></td>
<td>( 0^+, 1^+, 0^-, 1^-, 2^- , 2^+ )</td>
</tr>
<tr>
<td></td>
<td>( 0^+, 0^-, 2^-, 2^+, 1^+, 1^+ )</td>
</tr>
</tbody>
</table>

The condition (v) of Lemma 3 straightforwardly implies
Corollary 1 A feasible $S-$sequence, which starts with $s_1 = x^-$, $x \in \{0,1,\ldots,m\}$, contains at most one $y^+$, $y \in \{0,1,\ldots,m\}\{x\}$, such that $p_S(y^+) < p_S(y^-)$.

Obviously, every 1–unit robot cycle, if represented by an $S-$sequence, admits $2(m + 1)$ different representations, depending on which of $x^-$ or $x^+$, $x \in \{0,1,\ldots,m\}$, is chosen as the first entry ($s_1$) in corresponding $S$. We may easily achieve the uniqueness of $S-$sequence representation if we demand that the first element in a sequence should be the same for all $S-$sequences. Under this assumption, an active robot cycle admits the one and only $S-$sequence representation.

In what follows we will deal exclusively with $S-$sequences, in which

$$s_1 = 0^-.$$  \hspace{1cm}(14)

Lemma 4 There is a one-to-one correspondence between the set of all 1–unit active robot cycles and the set of all feasible sequences $S$ given by (1)-(3) and (14).

Proof. The proof is straightforward. Any robot cycle can be represented by a sequence of “load a machine” and “unload a machine” activities, that is, by an $S-$sequence. The condition (14) ascertains that each cycle translates into exactly one, unique $S-$sequence, the feasibility of which is by the definition. Conversely, due to Property 2, each feasible $S-$sequence corresponds to exactly one, unique active robot cycle. \hfill \Box

6 The number of active 1–unit cycles

In this Section, we devote our attention to the problem of identifying all distinct feasible $S-$sequences to represent the complete family of all active 1–unit cycles for the $m-$machine dual-gripper robotic flow shop producing identical parts.

6.1 Preliminaries

In this subsection, we introduce some additional notation and notions, which will be used in the subsequent analysis.

Let $K^+ = \{0^+,1^+,\ldots,m^+\}$ and $K^- = \{0^-,1^-,\ldots,m^-\}$.

Given a sequence $S = (s_1, s_2, \ldots, s_{2(m+1)})$, let $\chi_j^-(S)$ and $\chi_j^+(S)$ stand for the number of $s_j \in K^-$ entries and for the number of $s_j \in K^+$ entries, respectively, in $(s_1, s_2, \ldots, s_j)$ subsequence of $S$. For instance, given a sequence $S = (0^-,2^-,0^+,1^-,1^+,2^+)$, one would have
Furthermore, we introduce the notion of $T$–sequences.

In what follows, with sequence $S = (s_1, s_2, \ldots, s_{2(m+1)})$ we will associate a sequence

$$T = (t_1 t_2 \ldots t_{2(m+1)}), \quad t_j \in \{+,-\},$$

where

$$t_j = \begin{cases} +, & \text{if } s_j \in K^+; \\ - , & \text{if } s_j \in K^-, \end{cases}$$

for some $x$. In other words, given a sequence $S$, the sequence $T$ associated with $S$ is obtained from $S$ by substituting each $s_j \in K^+ \; (s_j \in K^-)$ in $S$ by mere $'+'$ ($'-'$) in $T$. For instance, sequence $S = (0^-, 2^-, 0^+, 1^-, 1^+, 2^+)$ translates into $T = (-,-,+,+-)$.

Obviously, each $S$–sequence associates with the unique $T$–sequence, whereas each $T$–sequence may correspond to a number of $S$–sequences. To illustrate the latter possibility, consider the sequence $T = (-,-,+,-,+,+)$. The possible feasible $S$–sequences with $s_1 = 0^-$, which associate with this $T$–sequence, are $S = (0^-, 1^-, 0^+, 2^-, 2^+, 1^+)$, $S = (0^-, 2^-, 0^+, 1^-, 1^+, 2^+)$, $S = (0^-, 1^-, 1^+, 2^-, 2^+, 0^+)$, $S = (0^-, 2^-, 2^+, 1^-, 1^+, 0^+)$, $S = (0^-, 1^-, 0^+, 2^-, 1^+, 2^+)$, and so on.

We now introduce some definitions and notation concerning $T$–sequences which we will further exploit in our work.

**Definition 4** The sequence $T = (t_1 t_2 \ldots t_{2n})$, $t_j \in \{+,-\}$, is called regular if each pair of its entries $t_{2j-1}$ and $t_{2j}$, $j = \overline{1,n}$, comprises exactly one “$-$” and one “$+$”.

Given a sequence $T = (t_1 t_2 \ldots t_{2n})$, $t_j \in \{+,-\}$, let $\chi_j^-(T)$ and $\chi_j^+(T)$ stand for the number of $'-'$ entries and $'+'$ entries, respectively, in $(t_1 t_2 \ldots t_j)$ subsequence of $T$. Then, trivially, the following fact takes place, by definition.

**Remark 4** In any regular sequence $T = (t_1 t_2 \ldots t_{2n})$, $\chi_{2j}^- = \chi_{2j}^+$ for any $j \in \{1,2,\ldots,n\}$.

Given a sequence $T = (t_1 t_2 \ldots t_{2n})$, $t_j \in \{+,-\}$, let a pair of its entries $t_{2j-1}$ and $t_{2j}$, $j = \overline{1,n}$, be called a duet. That is, we group consecutive entries of a $T$–sequence in pairs, so that a $T$–sequence
comprising $2n$ elements can be viewed as a series of $n$ successive duets. In what follows, we refer to the elements $t_{2j-1}$ and $t_{2j}$ as to the first and second elements, respectively, of duet $(t_{2j-1}, t_{2j})$, $j = 1, \ldots, n$.

Let $\alpha = (-+) \text{ and } \beta = (+-)$. Then a regular $T$-sequence can be viewed as a series of successive duets of form $\alpha$ and $\beta$ ($\alpha$-duets and $\beta$-duets), only.

**Definition 5** The sequence $T = (t_1t_2 \ldots t_{2n})$, $t_j \in \{+, -\}$, is called $\alpha$-regular if each pair of its entries $(t_{2j-1}t_{2j})$, $j = 1, \ldots, n$, is a duet of form $\alpha$ ($\alpha$-duet).

We are now ready to move to the main issue of identifying all possible 1–unit active cycles for an $m$-machine cell. Due to Lemma 4, the number of active 1–unit cycles is equal to the number of all feasible $S$-sequences satisfying (1)-(3) and (14). In what follows, we aim to identify all such feasible $S$-sequences.

The Corollary 1 along with condition (14) guarantee that for any feasible $S$-sequence under consideration there exists at most one $x \in \{1, 2, \ldots, m \}$ such that $p_S(x^+) < p_S(x^-)$. Accordingly, we split the discussion into two cases depending on the existence of such $x$. In each case we will proceed along the same lines. We will first establish the correspondence between the feasible $S$-sequences for the case under study and the $T$-sequences associated with them. We identify the structural properties of the corresponding $T$-sequences which may be converted into feasible $S$-sequences. We then prove that any arbitrary $T$-sequence of the required structure admits the conversion into a feasible $S$-sequence. We prove it by construction. Namely, in each case we will present an algorithm that actually constructs a feasible sequence $S$ from a given arbitrary $T$-sequence. Obviously, different $T$-sequences “generate” pairwise different $S$-sequences. And so, we may then substitute the problem of counting all feasible $S$-sequences by the following threefold problem: first, identify all possible $T$-sequences given by (15) for the case under study, and next, count all feasible $S$-sequences derived from each of these $T$-sequences (so that the condition (16) is satisfied), and finally, merely add the numbers up to get the desired number of all feasible $S$-sequences.

### 6.2 $S^1$–Sequences

Here we consider only those feasible $S$-sequences, in which

$$p_S(x^-) < p_S(x^+), \quad \text{for all } x \in \{0, 1, \ldots, m\}. \tag{17}$$
Thus, here we deal exclusively with $S$–sequences defined by (1)-(3), (14), and (17). For convenience, we refer to such sequences as to $S^1$–sequences. The aim is to count them all.

With a sequence $S^1$ under consideration we will associate a $T^1$–sequence as defined above by (15)-(16). Note that $t_1 = -$ and $t_{2(m+1)} = +$, due to (14) and (17). We first aim to establish the correspondence between the feasible $S^1$–sequences and the $T^1$–sequences associated with them.

Suppose that some feasible $S^1$–sequence is given. The assumptions (14), (17), along with the condition (i) of Lemma 3 guarantee that in any subsequence $(s_2, s_3, \ldots, s_j)$ with $j$ being an odd number $(3 \leq j \leq 2m + 1)$, the number of $s_i \in K^-$ is equal to the number of $s_i \in K^+$. Thus, in any subsequence $(t_2 t_3 \ldots t_j)$ of $T^1$ associated with $S^1$, with $j$ being an odd number, the number of $'-$ entries is equal to the number of $'+'$ entries. So, we deduce that, in $T^1$, any pair of $t_{2j}$ and $t_{2j+1}$, $j = 1, m$, comprises one “ − ” and one “ + ”. It straightforwardly implies

**Property 4** A sequence $T^1$ defined above must be of the following form:

$$T^1 = (-, \tilde{T}, +), \text{ where } \tilde{T} \text{ is a regular sequence.} \quad (18)$$

We now prove that any arbitrary $T^1$–sequence given by (15) and (18) admits the conversion into the feasible $S^1$–sequence. Below, we present an algorithm that constructs a feasible sequence $S^1$ from a given arbitrary $T^1$–sequence given by (15) and (18).

**Algorithm FindCycle$S^1$**

Input: An arbitrary sequence $T^1$ given by (15) and (18).
Output: A feasible sequence $S^1$ associated with $T^1$, so that condition (16) is satisfied.

**Step 1** Define $X_{new} := \{1, 2, \ldots, m\}$, $X_{old} := \{0\}$, $s_1 := 0^-$.  

**Step 2** For $j = 2$ to $2(m + 1)$:

**Step 2.1** If $t_j = -$ then
- take an arbitrary $x$ from $X_{new}$;
- assign $s_j := x^-$, set $X_{new} := X_{new} \setminus \{x\}$, $X_{old} := X_{old} \cup \{x\}$.

**Step 2.2** Otherwise (if $t_j = +$)
- take an arbitrary $x$ from $X_{old}$;
- assign $s_j := x^+$, set $X_{old} := X_{old} \setminus \{x\}$.

**Step 3** Output $S^1 = (s_1, s_2, \ldots, s_{2(m+1)})$. Stop.
We are now to show that the algorithm works, i.e., that it runs faultlessly to reach Step 3 and to output a feasible sequence $S^1$, for which conditions (16) and (17) are satisfied subject to a given $T^1$-sequence.

**Lemma 5** Given an arbitrary $T^1$-sequence, defined by (15) and (18), as input, Algorithm FindCycle $S^1$ outputs a feasible $S^1$-sequence associated with $T^1$.

Proof. Trivially, since each $'-$ entry of $T^1$ necessarily translates into $s_j \in K^-$, as well as each $'+'$ entry of $T^1$ necessarily goes into $s_j \in K^+$, the found $S^1$-sequence associates with $T^1$ by construction, i.e., condition (16) is satisfied. Furthermore, the operation of including $x$ into set $X_{old}$ is always preceded by $s_j := x^{-}$ assignment, for the same $x$ (Step 2.1). Thus, $s_j := x^{-}$ assignment always precedes $s_j := x^{+}$ assignment, for the same $x$. First of all, it implies that for $S^1$ the condition (17) is satisfied. Moreover, at the end of each execution of Step 2, for corresponding $j$, (and so, at the outset of each execution of Step 2 for $j + 1$), we always have

$$\chi_j^-(T^1) = \chi_j^+(T^1) + |X_{old}|.$$  \hspace{1cm} (19)

We now move to the issue of error-free workability of the algorithm, i.e., the successful run of the algorithm faultlessly resulting in reaching Step 3. Basically, we have to show that the choice of $x$ to fulfill the $s_j := x^{-}$ and $s_j := x^{+}$ assignments is always possible. Obviously, the assignment of $s_j \in K^-$, executed at Step 2.1, may not possibly cause any problems or failings. So, to prove the workability of the algorithm we only need to prove that the assignment of $s_j \in K^+$, executed at Step 2.2, also runs smoothly and faultlessly, namely, that at any time the operation “take an arbitrary $x$ from $X_{old}$” is to be executed, the set $X_{old}$ is not empty.

Consider the situation when Step 2.2 is to be run for some $j$, $2 \leq j \leq 2(m + 1)$. That is, we found out that $t_j = +$ and now some $x$ is to be taken from $X_{old}$. What we may have here then is one of the following two situations.

1. $j$ is even: $j = 2i$, for some $i$, and so, the $t_j$-entry of $T^1$-sequence is the first element of $'+-'$-duet ($\beta$-duet) in $\tilde{T}$-sequence. In this case, regularity of $\tilde{T}$ (Remark 4) and $t_1 = -$ condition imply that $\chi_{j-1}^-(T^1) = i$ and $\chi_{j-1}^+(T^1) = i - 1$, and thus, that $|X_{old}| = 1$, by (19).

2. $j$ is odd: $j = 2i + 1$, for some $i$, and, so, the $t_j$-entry of $T^1$-sequence is the second element of $'+'-'$-duet ($\alpha$-duet) in $\tilde{T}$-sequence. Then, due to regularity of $\tilde{T}$ (Remark 4) and $t_1 = -$ condition, we have that $\chi_{j-1}^-(T^1) = i + 1$ and $\chi_{j-1}^+(T^1) = i - 1$, and, therefore, $|X_{old}| = 2$, by (19).
Finally, for the last execution of Step 2.2, i.e., for $j = 2(m + 1)$, $t_j = +$, we obviously have $|X_{old}| = \chi_2^{(m+1)}(T^1) - \chi_2^{(m+1)}(T^1) = 1$ (by Remark 4, $t_1 = -$ condition and (19)). So, whenever Step 2.2 is to be run, the set $X_{old}$ is not empty.

Thus, the workability of the algorithm is established. The only issue which is left to be justified is the feasibility of the sequence $S^1$, as it is. That is, we have to demonstrate that sequence $S^1$ satisfies the Lemma 3. Naturally, we are concerned with the requirement (i) of Lemma only since the necessity in four other requirements is ruled out by (17). The situation described in this requirement is conditioned by $p_S(x^-) < p_S(y^-) < \min\{p_S(x^+), p_S(y^+)\}$. Let’s analyze the behavior of the algorithm. If such a situation takes the place, we would have the following. For some $j$ at Step 2 we have had $t_j = -$ and assignment $s_j := y^-$ has been made, with the assignment $s_i := x^-$ being executed previously for some $i < j$ and some $x$, whereas $x^+ \notin \{s_l\}_{l \leq j}$. It implies that after execution Step 2 for this particular $j$ we have $\{x, y\} \subseteq X_{old}$. The fact of $|X_{old}| \leq 2$ yields $X_{old} = \{x, y\}$, and, together with (19) implies also that $t_{j+1} = +$. Thus, for $j + 1$, Step 2.2 is to be run, and either $s_{j+1} := x^+$ or $s_{j+1} := y^+$ is to be made, and, hence, the requirement (i) of Lemma 3 is satisfied. Hence, the feasibility of $S^1$ is assured. This proves the Lemma. □

We now turn to the issue of identifying all feasible $S^1$–sequences.

**Lemma 6** The number of all pairwise different $S^1$–sequences is equal to

$$m! \times \sum_{R=0}^{m} (2^R \times C^R_m),$$

(20)

where $C^R_m = \frac{m!}{R!(m-R)!}$.

Proof. The workability of Algorithm FindCycle$S^1$ for any arbitrary $T^1$–sequence of form (15) and (18) as the algorithm’s input proves that any arbitrary $T^1$–sequence given by (15) and (18) produces a set of feasible $S^1$–sequences. Furthermore, it is also easy to see that, if all possible choices of taking an arbitrary $x$ on all steps of Algorithm are exhausted, we come up with the set comprising all possible feasible $S^1$–sequences, which may be “generated” from a given $T^1$–sequence. Thus, the problem of counting all feasible $S^1$–sequences derived from a $T^1$–sequence reduces to that of counting all possible ways of taking an arbitrary $x$ on Steps 2.1–2.2 of Algorithm FindCycle$S^1$.

The first straightforward observation is as follows: the “conversion” of all ‘$-$’ in $T^1$ to $s_j = x^-$ is made in an arbitrary sequential manner, no restrictions whatever, namely, the values of $x$ are consecutively taken from the set $\{1, 2, \ldots, m\}$ in an arbitrary order. The number of permutations of an $m$–set is $m!$; so, given a $T^1$–sequence, there exist $m!$ possible different ways to assign corresponding $s_j \in K^-$. 

18
Assume that a particular assignment of all \( s_j \in K^- \) is chosen. The assignment of \( s_j \in K^+ \) is carried out on Step 2.2 of algorithm. As shown in proof of Lemma 5, the only situation, when the choice of \( x \) to fulfill \( s_j := x^+ \) assignment is not unique, is when the corresponding the \( t_j \)-entry of \( T^1 \)-sequence is the second element of \( \alpha \)-duet in \( \tilde{T} \)-sequence. With \( |X_{old}| = 2 \) in this case, we have then the freedom of choice between two candidates. Thus, each of \( \alpha \)-duets in \( \tilde{T} \)-sequence generates two possibilities for \( s_j := x^+ \) assignment. Consequently, if a given \( T^1 \)-sequence is to be converted into a feasible \( S^1 \)-sequence subject to a particular, predefined assignment of \( s_j \in K^- \), there exist \( 2^R \) possible different ways to assign the remaining \( s_j \), i.e., \( s_j \in K^+ \), where \( R \) is the number of \( \alpha \)-duets in \( \tilde{T} \)-sequence. Thus, every \( T^1 \)-sequence with \( R \) of \( \alpha \)-duets in its \( \tilde{T} \)-subsequence may correspond to one of \( 2^R \times m! \) different feasible \( S^1 \)-sequences.

The only issue left to be resolved is the issue of identifying all the different \( T^1 \)-sequences, or what is the same, all the different \( \tilde{T} \)-sequences. As a regular sequence, a \( \tilde{T} \)-sequence is a series of \( \alpha \)- and \( \beta \)-duets, with the total number of all \( \alpha \)- and \( \beta \)-entries being equal to \( m \). Evidently, the number of different \( \tilde{T} \)-sequences with exactly \( R \) occurrences of \( \alpha \)-duets, for some \( R \), \( 0 \leq R \leq m \), is \( C^R_m \). So, there are \( C^R_m \) different \( T^1 \)-sequences with \( R \) occurrences of \( \alpha \)-duets in the \( \tilde{T} \)-subsequence, each of them corresponds to \( 2^R \times m! \) different feasible \( S^1 \)-sequences, whereas \( R \) varies from 0 to \( m \). To obtain the number of all possible feasible \( S^1 \)-sequences, we merely sum the numbers up to get: \( \sum_{R=0}^{m} (C^R_m \times 2^R \times m!) \), which can be rewritten as (20). □

6.3 S^2-Sequences

In this subsection, we turn our attention to the case of all those feasible \( S \)-sequences, in which there exists exactly one \( y \in \{1, 2, \ldots, m\} \) such that \( p_S(y^+ < p_S(y^-) \), that is

\[
p_S(y^-) > p_S(y^+), \text{ for one } y \in \{1, \ldots, m\},
\]

(21)

\[
p_S(x^-) < p_S(x^+), \text{ for all } x \in \{0, 1, \ldots, m\}\setminus\{y\}.
\]

(22)

Thus, here we deal exclusively with \( S \)-sequences defined by (1)-(3), (14), (21)-(22). We refer to such sequences as to \( S^2 \)-sequences. Again, we aim to identify them all.

We will proceed along the same lines as in Subsection 6.2.

Similarly, with a sequence \( S^2 \) we will associate a \( T^2 \)-sequence as defined above by (15)-(16). Note that \( t_1 = - \), due to (14). Again, we aim to establish the correspondence between the set of all feasible
$S^2$–sequences and all possible $T^2$–sequences, associated with these $S^2$–sequences. We start with the analysis of the possible structure of a sequence $S^2$.

Suppose, we are given some feasible $S^2$–sequence.

In $S^2$, we then can identify the two two-elements subsequences $(s_{2u-1}, s_{2u})$ and $(s_{2v-1}, s_{2v})$, $1 \leq u \leq v \leq m + 1$, $v \neq 1$, such that $y^+ \in (s_{2u-1}, s_{2u})$ and $y^- \in (s_{2v-1}, s_{2v})$. Then the following observations hold true.

**Property 5** A feasible $S^2$–sequence satisfies the following conditions:

1) The subsequence $(s_1, s_2, \ldots, s_{2(u-1)})$ must be of the form

$$(s_1, s_2, \ldots, s_{2(u-1)}) = (0^-, 0^+, x_1^-, x_1^+, \ldots, x_{u-1}^-, x_{u-1}^+),$$

where $x_j \in \{1, 2, \ldots, m\}\{y\}$, $x_j$ are pairwise different. Furthermore, $\chi_{2(u-1)}^{-}(S^2) = \chi_{2(u-1)}^{+}(S^2)$.

2) If $u = 1$, one must have $(s_{2u-1}, s_{2u}) = (s_1, s_2) = (0^-, y^+)$. Otherwise, if $u > 1$, the subsequence $(s_{2u-1}, s_{2u})$ comprises $y^+$ and $x_u^-$, for some $x_u \in \{1, 2, \ldots, m\}\{y\}$. Furthermore, if $v = u > 1$ then $(s_{2u-1}, s_{2u}) = (y^-, y^+)$. $\chi_{2u}^{-}(S^2) = \chi_{2u}^{+}(S^2)$.

3) In any subsequence $(s_{2u+1}, s_{2u+2}, \ldots, s_{2j})$, $u + 1 \leq j \leq v - 1$, the number of $s_i \in K^-$ is equal to the number of $s_i \in K^+$, Moreover for any $s_i = x^+ \in (s_{2u+1}, s_{2u+2}, \ldots, s_{j})$, $2(u + 1) \leq j \leq 2(v - 1)$ there exists $s_l = x^- \in (s_{2u-1}, s_{2u}, \ldots, s_{j})$ such that $l < i$. Furthermore, $\chi_{2j}^{-}(S^2) = \chi_{2j}^{+}(S^2)$, $u + 1 \leq j \leq v - 1$.

4) The subsequence $(s_{2v-1}, s_{2v})$ comprises $y^-$ and $x_v^+$, for some $x_v \in \{1, 2, \ldots, m\}\{y\}$, such that $x_v^- \in (s_{2u-1}, s_{2u}, \ldots, s_{2v-1})$. Furthermore, $\chi_{2v}^{-}(S^2) = \chi_{2v}^{+}(S^2)$, and also, for any $x$ such that $x^- \in \{s_1, s_2, \ldots, s_{2v}\}$, one must have $x^+ \in \{s_1, s_2, \ldots, s_{2v}\}$ as well.

5) The subsequence $(s_{2v+1}, s_{2v+2}, \ldots, s_{2(m+1)})$ must be of the form

$$(s_{2v+1}, s_{2v+2}, \ldots, s_{2(m+1)}) = (x_1^-, x_1^+, x_2^-, x_2^+, \ldots, x_{m-v+1}^-, x_{m-v+1}^+),$$

where $x_j \in \{1, 2, \ldots, m\}\{y\}$, $x_j$ are pairwise different. Furthermore, $\chi_{2j}^{-}(S^2) = \chi_{2j}^{+}(S^2)$, $v + 1 \leq j \leq m$.

Proof. 1): due to (14), (22), and the condition (ii) of Lemma 3;
2): from (22) and the above item 1) of this Property;
3): from (22), the above item 2) of this Property, and condition (i) of Lemma 3;
We now can identify a possible structure of $T^2$, associated with the sequence $S^2$. The Property above straightforwardly implies that

$$T^2 = \left( T', \pi_u, \tilde{T}, \pi_v, T'' \right),$$

(23)

$$T' = (t_1 t_2 \ldots t_{2(u-1)}) \quad \text{and} \quad T'' = (t_{2u+1} t_{2u+2} \ldots t_{2(m+1)}) \quad \text{are } \alpha \text{ -- regular;}$$

$$\pi_u = (t_{2u-1} t_{2u}) \quad \text{is} \quad \begin{cases} (-+) & \text{if } u = 1 \\ (-+) & \text{if } u = v, \ 1 < u \leq m; \end{cases}$$

$$\pi_v = (t_{2u-1} t_{2u}) \quad \text{is either} \quad (-+) \quad \text{or} \quad (+-), \quad u < v \leq m;$$

$$\tilde{T} = (t_{2u+1} t_{2(u+1)} \ldots t_{2u-1}) \quad \text{is regular.}$$

And this, in turn, implies the following

**Property 6** An arbitrary $T^2$--sequence described by (15) and (23)-(24) may be uniquely defined by specifying a sequence $(\pi_u, \tilde{T}, \pi_v)$, and setting the values of $u$ and $v$.

We next prove that any arbitrary $T^2$--sequence given by (15) and (23)-(24) admits the conversion into the feasible $S^2$--sequence. Analogously to the Subsection 6.2, we prove it by construction, that is, we present an algorithm that actually constructs a feasible sequence $S^2$ from a given arbitrary $T^2$--sequence given by (15) and (23)-(24).

**Algorithm FindCycle**

Input: An arbitrary sequence $T^2$ given by (15) and (23)-(24).
Output: A feasible sequence $S^2$ associated with $T^2$, so that conditions (16) is satisfied.

**Step 1** Define $X_{new} := \{1, \ldots , m\}$, $X_{old} := \emptyset$; $s_1 := 0^-$

**Step 2** Take an arbitrary $x$ from $X_{new}$; assign $y := x$; set $X_{new} := X_{new}\backslash \{x\}$

**Step 3** If $u = v$ then

**Step 3.1** Assign $s_2 := 0^+$

**Step 3.2** For $j = 2$ to $u - 1$:

- take an arbitrary $x$ from $X_{new}$; assign $s_{2j-1} := x^-$, $s_{2j} := x^+$; set $X_{new} := X_{new}\backslash \{x\}$

**Step 3.3** Assign $s_{2u-1} := y^+$, $s_{2u} := y^-$

**Step 3.4** For $j = u + 1$ to $m + 1$:

- take an arbitrary $x$ from $X_{new}$; assign $s_{2j-1} := x^-$, $s_{2j} := x^+$; set $X_{new} := X_{new}\backslash \{x\}$

**Step 3.5** Go to Step 5
Step 4 Otherwise \((u \neq v)\)

\[\text{Step 4.1} \quad \text{If } u = 1 \text{ then}
\]
\[
\begin{align*}
\text{assign } s_2 & := y^+, \ X_{\text{old}} := \{0\}; \\
\text{go to Step 4.3}
\end{align*}
\]

\[\text{Step 4.2} \quad \text{Otherwise } (u > 1)
\]

\[\text{Step 4.2.1} \quad \text{Assign } s_2 := 0^+
\]

\[\text{Step 4.2.2} \quad \text{For } j = 2 \text{ to } u - 1:
\]
\[
\begin{align*}
\text{take an arbitrary } x & \text{ from } X_{\text{new}}, \text{ assign } s_{2j - 1} := x^-, \ s_{2j} := x^+, \text{ set } X_{\text{new}} := X_{\text{new}} \setminus \{x\}.
\end{align*}
\]

\[\text{Step 4.2.3} \quad \text{For } j = 2 \text{ to } u - 1:
\]
\[
\begin{align*}
\text{If } t_j & = - \text{ then}
\end{align*}
\]
\[
\begin{align*}
\text{take an arbitrary } x & \text{ from } X_{\text{new}}; \\
\text{assign } s_j & := x^-; \text{ set } X_{\text{new}} := X_{\text{new}} \setminus \{x\}; \ X_{\text{old}} := X_{\text{old}} \cup \{x\}
\end{align*}
\]
\[
\begin{align*}
\text{Otherwise } (t_j = +) & \text{ assign } s_j := y^+
\end{align*}
\]

\[\text{Step 4.3} \quad \text{For } j = 2u + 1 \text{ to } 2(v - 1):
\]

\[\text{Step 4.3.1} \quad \text{If } t_j = - \text{ then}
\]
\[
\begin{align*}
\text{take an arbitrary } x & \text{ from } X_{\text{new}}; \\
\text{assign } s_j & := x^-; \text{ set } X_{\text{new}} := X_{\text{new}} \setminus \{x\}, \ X_{\text{old}} := X_{\text{old}} \cup \{x\}
\end{align*}
\]

\[\text{Step 4.3.2} \quad \text{Otherwise } (t_j = +)
\]
\[
\begin{align*}
\text{take an arbitrary } x & \text{ from } X_{\text{old}}; \\
\text{assign } s_j & := x^+; \text{ set } X_{\text{old}} := X_{\text{old}} \setminus \{x\}
\end{align*}
\]

\[\text{Step 4.4} \quad \text{For } j = 2v - 1 \text{ to } 2v
\]
\[
\begin{align*}
\text{If } t_j & = + \text{ then}
\end{align*}
\]
\[
\begin{align*}
\text{take an arbitrary } x & \text{ from } X_{\text{old}}; \text{ assign } s_j := x^+; \text{ set } X_{\text{old}} := X_{\text{old}} \setminus \{x\}
\end{align*}
\]
\[
\begin{align*}
\text{Otherwise } (t_j = -) & \text{ assign } s_j := y^-
\end{align*}
\]

\[\text{Step 4.5} \quad \text{For } j = v + 1 \text{ to } m + 1:
\]
\[
\begin{align*}
\text{take an arbitrary } x & \text{ from } X_{\text{new}}, \text{ assign } s_{2j - 1} := x^-, \ s_{2j} := x^+, \text{ set } X_{\text{new}} := X_{\text{new}} \setminus \{x\}
\end{align*}
\]

\[\text{Step 5} \quad \text{Output } S^2 = (s_1, s_2, \ldots, s_{2(m+1)}). \text{ Stop.}
\]

We are now to show that the algorithm indeed works, i.e., that it always successfully reaches Step 5 and outputs a feasible sequence \(S^2\) which satisfies the conditions (16) and (21)-(22).

**Lemma 7** Given an arbitrary \(T^2\) sequence, defined by (15) and (23)-(24), as input, Algorithm \(\text{FindCycle}(S^2)\) outputs a feasible \(S^2\)-sequence associated with \(T^2\).

Proof. Trivially, the assigning of \(s_j := y^+\), as well as of all \(s_j \in K^-\) may not possibly cause any problems. Furthermore, in case of \(u = v\) the assignment of all \(s_j \in K^+ \setminus \{y^+\}\) does not pose any ambiguity, as well. If \(u < v\), the assignment of \(s_j \in K^+ \setminus \{y^+\}\) performed on Steps 4.2.1-4.2.2 for \(j \leq 2(u - 1)\), as well as on Step 4.5 - for \(j \geq 2v + 1\) is obviously unique. We also observe that at the outset of the first execution of Step 4.3 we always have \(|X_{\text{old}}| = 1\) (from Steps 4.1 or 4.2.3, whichever has been executed). Precisely
the same arguments as in analysis of Algorithm FindCycle show that every time Step 4.3.2 is to be executed we have \( 1 \leq |X_{old}| \leq 2 \). And so, every time when the operation “take an arbitrary \( x \) from \( X_{old} \)” is to be performed, the set \( X_{old} \) is not empty. Finally, upon completion of Step 4.3 (at the outset of Step 4.4) we also necessarily have \( |X_{old}| = 1 \), and, therefore, the assignment \( s_j := x^+ \) carried out on Step 4.4 is also trouble-free. Thus, the algorithm runs error-free.

Furthermore, the conditions (16) and (21)-(22) are satisfied by construction. Moreover, it is easy to show that the found \( S^2 \)-sequence satisfies all the conditions of Property 5, and then that the requirements (i)-(iv) of Lemma 3 are obligatory met, and therefore the feasibility of \( S^2 \)-sequence is ascertained. This proves the Lemma.

We now move to the issue of identifying all feasible \( S^2 \)-sequences.

**Lemma 8** The number of all pairwise different \( S^2 \)-sequences is equal to

\[
m! \times \left[ m + \sum_{Q=0}^{m-1} \left( 2 + 4(m - Q - 1) \times \sum_{R=0}^{Q} (2^R \times C^R_Q) \right) \right],
\]

where \( C^R_Q = \frac{Q!}{R!(Q-R)!} \).

Proof. We first note that the workability of the algorithm for any arbitrary \( T^2 \)-sequence, given by (15) and (23)-(24), as the algorithm’s input proves that any arbitrary \( T^2 \)-sequence, given by (15) and (23)-(24), produces a set of feasible \( S^2 \)-sequences. Furthermore, for a given \( T^2 \)-sequence, having exhausted all possible choices of taking an arbitrary \( x \) on all steps of the algorithm, we evidently come up with the set of all possible feasible \( S^2 \)-sequences associated with \( T^2 \). Thus, the problem of counting all feasible \( S^2 \)-sequences derived from a \( T^2 \)-sequence reduces to that of counting all possible ways of taking an arbitrary \( x \) on Steps 2-4 of Algorithm FindCycle\( S^2 \).

Well, there are \( m \) possibilities to chose \( y \) from set \( \{1, 2, \ldots, m\} \). Furthermore, the conversion of all ‘−’ in \( T^2 \) to \( s_j = \{x^- | x^- \in K^- \setminus \{y^-\}\} \) is performed with no restriction on choice of \( x \): the values of \( x \) are consecutively taken from the set \( \{1, 2, \ldots, m\}\setminus\{y\} \) in an arbitrary order. Hence, the number of possible different ways to make the selection of \( xs \) to fulfill \( s_j := x^-, x^- \neq y^- \), assignment is then \( (m - 1)! \).

Assume that a particular \( y \in \{1, 2, \ldots, m\} \) and the assignment of \( s_j \in K^- \setminus \{y^-\} \) is chosen (as one particular choice out of \( m! \) possibilities). Then the assignment of \( s_j = y^- \) is the unique, and so what is left to be considered is the assignment of all \( s_j \in K^+ \setminus \{y^+\} \). We split the discussion into two cases.

The first case concerns the situation when Step 3 of the algorithm is run \((u = v)\). Clearly, here the assignment of all \( s_j \in K^+ \setminus \{y^+\} \) is the unique. So, the total number of possible \( S^2 \)-sequences derived from a given \( T^2 \)-sequence with \( u = v \) is equal to \( m! \).
Now assume that Step 4 of the Algorithm is run \((u < v)\). The assignment of \(s_j \in K^+ \setminus \{y^+\}, j \leq 2(u - 1)\) and \(j \geq 2v + 1\), is carried out on Steps 4.2.2 and 4.5 of the algorithm, and it is unique (the choice of \(x\) to fulfill the assignment \(s_j = x^+\) is predetermined by the previously made assignment \(s_j = x^-\) for the same \(x\)). At Step 4.4, for \(t_j = +\) in \(\pi_v\) we have \(|X_{\text{old}}| = 1\) and thus the choice of \(x\) to make the corresponding assignment \(s_j := x^+\) is unique.

So, what is left to be considered is the assignment of \(s_j \in K^+ \setminus \{y^+\}, 2u + 1 \leq j \leq 2v - 1\), which is performed on Step 4.3. We observe that after completion the Step 4.2 we have \(j \leq 2\pi\) equal to 2, \(\tilde{\pi}\) is unique. 

The alignment of Algorithm FindCycle\(^1\) is fully defined if described as a collection of all pairs of \(\{\alpha\} \) and \((23)-(24)\) is fully defined if described as a collection of all pairs of \(\alpha\)-values. Since each duet \(\pi\) in \(\tilde{T}\)-sequence. Hence, if \(u < v\), every \(T^2\)-sequence with \(R\) of \(\alpha\)-duets in its \(T\)-subsequence corresponds to \(m! \times 2^R\) of different feasible \(S^2\)-sequences.

We now turn to the issue of different \(T^2\)-sequences. Property 6 implies that the set of all \(T^2\)-sequences of form (15) and (23)-(24) is fully defined if described as a collection of all pairs of \(u\) and \(v\) values, taken together with each of possible \((\pi_u, \tilde{T}, \pi_v)\) sequences for this particular pair of \(u\) and \(v\). Evidently, different \(T^2\)-sequences, if described so, produce the different \(S^2\)-sequences, no two \(S^2\)-sequences, generated from different \(T^2\)-sequences, are the same. We aim to identify all possible pairwise different \(T^2\)-sequences.

If \(u = v\), \(\pi_u = \pi_v\), \(\tilde{T}\) is empty, and so we have \(m\) possible different \(T^2\)-sequences subject to the different possible values of \(u = v \in \{2, 3, \ldots m + 1\}\), and so the number of all feasible \(S^2\)-sequences associated with all such \(T^2\)-sequences is equal to \(m \times m!\).

Now suppose \(u < v\). Let \(Q\) be the number of duets in \(\tilde{T}\). That is, \(Q\) is the number of duets separating \(\pi_u\) and \(\pi_v\) in \(T^2\), i.e., \(Q = v - u - 1\). Using the same arguments as in analysis of Algorithm FindCycle\(^1\), we infer that for a given value of \(Q\), there exist \(C^R_Q\) different \(\tilde{T}\)-subsequences of \(T^2\), where \(R\) is the number of \(\alpha\)-duets in \(\tilde{T}\).

If \(u = 1\), the possible value of \(Q\) varies from 0 to \(m - 1\), depending solely on the value of \(v\), the duet \(\pi_u\) is unique, the duet \(\pi_v\) may take either \(\gamma^-\) or \(\gamma^+\)-form. Thus each of \(\sum_{Q=0}^{m-1} \sum_{R=0}^{Q} C^R_Q\) different \(\tilde{T}\)-sequences admits 2 possibilities for the duet \(\pi_v\). And, so, the number of possible \(T^2\)-sequences is equal to \(2 \times \sum_{Q=0}^{m-1} \sum_{R=0}^{Q} C^R_Q\). Hence, the number of all possible feasible \(S^2\)-sequences, which come from \(T^2\)-sequences with \(u = 1\), is equal to \(m! \times 2 \sum_{Q=0}^{m-1} \sum_{R=0}^{Q} 2^R \times C^R_Q\).

Otherwise, if \(u > 1\), the value of \(Q\) may vary from 0 to \(m - 2\). Furthermore, for a particular \(Q\), there are \((m - Q - 1)\) ways to assign different pairs of \(u\) and \(v\)-values. Since each duet \(\pi_v\) and \(\pi_u\) may
Table 1: Number $N$ of 1-unit cycles for various values of $m$.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>46</td>
</tr>
<tr>
<td>3</td>
<td>456</td>
</tr>
<tr>
<td>4</td>
<td>5 688</td>
</tr>
<tr>
<td>5</td>
<td>86 640</td>
</tr>
<tr>
<td>6</td>
<td>1 568 880</td>
</tr>
<tr>
<td>7</td>
<td>33 022 080</td>
</tr>
<tr>
<td>8</td>
<td>793 215 360</td>
</tr>
<tr>
<td>9</td>
<td>2 1 423 7 09 440</td>
</tr>
<tr>
<td>10</td>
<td>64 2 787 4 88 000</td>
</tr>
</tbody>
</table>

Table 1 shows the growth of $N$ for various values of $m$.

take either $' - +'$ or $' + -'$-form, each of $(m - Q - 1) \times \sum_{R=0}^{Q} C_{Q}^{R}$ different $\tilde{T}$-sequences admits 4 possibilities for the duets $\pi_{u}$ and $\pi_{v}$. And, so, the number of possible $T^{2}$-sequences for a given $Q$ is equal to $4(m - Q - 1) \times \sum_{R=0}^{Q} C_{Q}^{R}$, each of them generating $2^{R}$ different $S^{2}$-sequences. Summing up the numbers for all possible values of $Q$, we derive the total number of all different $S^{2}$-sequences, generated by the set of all different $T^{2}$-sequences with $u > 1$, $u \neq v$:

$$m! \times \sum_{Q=0}^{m-2} 4(m - Q - 1) \times \sum_{R=0}^{Q} (2^{R} \times C_{Q}^{R}).$$

Noticing that for $Q = m - 1$, we have $4(m - Q - 1) \times \sum_{R=0}^{Q} C_{Q}^{R} = 0$, and adding up the obtain numbers of $S^{2}$-sequences for all three possible situations considered ($u = v$, $u = 1$, $1 < u < v$) we obtain the desired formula (25). □

6.4 The Result

The subsections 6.1-6.3 straightforwardly imply the following

**Theorem 1** For the $RF_{m}^{2}|q = 1|C_{t}$ problem, the number $N$ of active 1–unit cycles is equal to

$$N = m! \times \left[ m + \sum_{R=0}^{m} (2^{R} \times C_{m}^{R}) + \sum_{Q=0}^{m-1} \left( (2 + 4(m - Q - 1)) \times \sum_{R=0}^{Q} (2^{R} \times C_{Q}^{R}) \right) \right],$$

where $C_{u}^{w} = \frac{w!}{w!(u-w)!}$. 

Table 1 shows the growth of $N$ for various values of $m$. 

25
7 A special case of an $m$-machine dual-gripper robot cell

In this section, we consider a special case of an $m$-machine robotic cell with a dual-gripper robot to produce a set of identical parts. Namely, we pose an assumption that $\max\{\theta, \theta_t\} \leq \delta$. In practical situations, this problem variant is in frequent occurrence, and possibly, reflects the most common case in real-life robotic systems.

Surprisingly enough, it turns out that, at least in case of 1-unit cycles, the $\max\{\theta, \theta_t\} \leq \delta$ assumption drastically simplifies the problem. In what follows, we will show that the problem of finding an optimal 1-unit cycle for $RF_2^m | q = 1 | C_t$ problem with $\max\{\theta, \theta_t\} \leq \delta$ is efficiently solvable.

We use the problem notation as given in Section 3. For convenience, we define the set of machine indices:

$$I = \{1, 2, \ldots, m\}. \quad (26)$$

We also exploit $S-$sequences notation (1)-(3) to represent a robot cycle, though with one slight modification. Namely, to avoid unnecessary confusion, instead of $x^-$ and $x^+$, $x \in \{0, 1, \ldots, m\}$, in what follows we will write $[x]^-$ and $[x]^+$, respectively.

We start with deriving the general lower bound on the optimal solution. We let

$$I_\theta = \{k|p_k \leq \theta, 1 \leq k \leq m\}, \quad \bar{T}_\theta = I \setminus I_\theta, \quad r = |\bar{T}_\theta|. \quad (27)$$

**Lemma 9** For $RF_2^m | q = 1 | C_t$ problem with $\max\{\theta, \theta_t\} \leq \delta$, the following lower bound on the length $T(C)$ of any 1-unit cycle $C$ holds true

$$T(C) \geq \max \left\{ \frac{\sum_{k \in I_\theta} p_k + \theta + 2\epsilon}{\delta} \left| \sum_{k \in I_\theta} p_k + r\theta + (m + 1)\delta + 2(m + 1)\epsilon \right| \right\}. \quad (28)$$

Proof. Let $C$ be an arbitrary 1-unit robot cycle, represented by a sequence of “load a machine” and “unload a machine” operations, and let sequence $S$ be its notational representation. Consider any machine $M_k$, $k = 1, 2, \ldots, m$, and let $\tau_k$ be the time taken by (partial) execution of cycle $C$ from the moment when robot is just to start unloading machine $M_k$ and until the moment when the robot just finished loading machine $M_k$. Trivially, $T(C) \geq \tau_k + p_k$. Furthermore, if in $C$ the operation “load machine $M_k$” follows the operation “unload machine $M_k$” immediately (there are no other operations executed in-between), then, straightforwardly, $\tau_k = \epsilon + \theta + \epsilon$. Otherwise, there is at least one operation executed after “unload $M_k$” operation and before “load $M_k$”. Obviously, such an operation would be
Given cycle $C$ and its notational representation by sequence $S = (s_1, s_2, \ldots, s_{2(m+1)})$, by a contribution $\Delta_{s_j}$ to the cycle time $T(C)$ for a particular $s_j$, $j = 1, 2, \ldots, 2(m+1)$, we will mean the time taken by a partial cycle execution between the point when the robot has just completed operation $s_{j-1}$ and is just about to start the execution of operation $s_j$. Note that the operations $s_j$ are nothing but “load a machine” or “unload a machine” operations, which take $\epsilon$ time each. The expression for a cycle time is then

$$T(C) = \sum_{j=1}^{2(m+1)} (\Delta_{s_j} + \epsilon) = \sum_{j=1}^{2(m+1)} \Delta_{s_j} + 2(m+1)\epsilon. \quad (30)$$

In sequence $S$, each of its entries $s_j \in \{[k]^-, [k]^+ | 1 \leq k \leq m\}$, $j = 1, 2, \ldots, 2(m+1)$, contributes to the cycle time as shown in Table 2. We remark, that the sign “$\geq$” in the estimations $\Delta_{s_j}$, as given in Table 2, allows for the robot waiting times at machines as well as for the robot travel time between non-consecutive machines, if such occur.

Let $I'$ and $I''$ denote the set of indices $k$ which satisfy the condition (i) and the condition (ii) in Table 2, respectively. Formally, we have

$$I' = \{ k \ | p_S([k]^-) = p_S([k-1]^+) + 1, 1 \leq k \leq m \}$$

and

$$I'' = \{ k \ | p_S([k]^+) = p_S([k+1]^-) + 1, 1 \leq k \leq m \}.$$
Then we have the following:

\[
2(m+1) \sum_{j=1}^{\Delta_s j} \geq \Delta_{[0]}^- + \Delta_{[m]}^+ + \sum_{k \in I'} p_{k-1} + \sum_{i=1}^{I'} \delta + \sum_{i=1}^{|I'|} \theta + \sum_{i=1}^{|I'|} \delta.
\]  

(31)

We first observe that \(\Delta_{[0]}^- + \Delta_{[m]}^+ \geq \delta\) since the execution of at least one of operations \([0]^-\) and \([m]^+\) necessarily contributes to \(T(C)\) the robot travel time between some machine and \(I/O\) hopper. Furthermore, it is easy to see that for each \(j\) such that \(s_j = [k]^-\) with \(k \in I'\) one must have \(s_{j-1} = [k-1]^+\) with \((k-1) \notin I''\). Consequently, \(|I'| \leq |I'\setminus I''|\). And so, (31), condition \(\theta \leq \delta\) and the above observations yield

\[
\sum_{j=1}^{2(m+1)} \Delta_s j \geq \delta + \sum_{k \in I'} p_{k-1} + \sum_{i=1}^{|I'|} \theta + \sum_{i=1}^{|I'|} \delta =
\]

\[
= \delta + \sum_{k \in I'} p_{k-1} + m\delta + \sum_{i=1}^{|I'|} \theta + \sum_{i=1}^{|I'|\setminus I''} \delta \geq
\]

\[
\geq (m+1)\delta + \sum_{k \in I'} p_{k-1} + \sum_{i=1}^{|I'|\setminus I''} \theta = (m+1)\delta + \sum_{k \in I'} p_{k-1} + \sum_{i=1}^{|I'|\setminus I''} \theta =
\]

\[
\geq (m+1)\delta + \sum_{k \in I'} \min \{p_k, \theta\} = (m+1)\delta + \sum_{k \in I} p_k + \sum_{i=1}^{|I|} \theta.
\]

The above result taken together with (30) and (27) yields

\[
T(C) \geq \sum_{k \in I} p_k + r\theta + (m+1)\delta + 2(m+1)e.
\]

(32)

Combining the relations (29) and (32) gives the desired bound (28). □

We are now ready to demonstrate that the problem of finding an optimal 1–unit cycle for \(RF^2_m |q = 1| C_t\) problem is efficiently solvable provided that \(\max \{\theta, \theta_t\} \leq \delta\), the optimality of the found 1–unit robot cyclic strategy being guaranteed by the fact that its cycle length is in fact equal to the lower bound (28).

**Algorithm OPT1**

**INPUT:** The problem instance of \(RF^2_m |q = 1| C_t\) problem. The sets \(I_\theta\) and \(T_\theta\) are as defined by (27).

**OUTPUT:** A 1–unit cycle \(C^*\).

**Initial setting:** Machines \(M_k, k \in T_\theta\), are occupied with a part; machines \(M_k, k \in I_\theta\), are empty. The robot is positioned at \(I/O\)-station; both robot grippers are empty.

**Step 1** The robot picks up a part from Input buffer at \(I/O\)-station.

**Step 2** For \(k\) from 1 to \(m\):

**Step 2.1** The robot moves to machine \(M_k\).
Step 2.2 If $k \in I_\theta$ then
- if necessary, the robot waits for $w_k$ time units for a part on $M_k$ to complete processing;
- the robot unloads a part from $M_k(\epsilon)$;
- the robot switches grippers ($\theta$);
- the robot loads a part on $M_k(\epsilon)$.

Step 2.3 Otherwise ($k \notin I_\theta$)
- the robot loads a part on $M_k(\epsilon)$;
- the robot waits for $p_k$ time units for a part on $M_k$ to be processed;
- the robot unloads a part from $M_k(\epsilon)$.

Step 3 The robot moves to $I/O$ hopper; drops a part at Output device of $I/O$-station.

The order of complexity of the above algorithm is $O(m)$. The theorem below analyses its performance.

Theorem 2 For $RF^2_m | q = 1 | C_t$ problem with $\max\{\theta, \theta_t\} \leq \delta$, Algorithm OPT1 produces an optimal 1–unit cycle $C^*$. The cycle time of $C^*$ is

$$T(C^*) = \max \left\{ \max_{k \in I_\theta} p_k + 2\epsilon + \theta; \sum_{k \in I_\theta} p_k + \sum_{i=1}^{\lceil T_\theta \rceil} \theta + (m+1)\delta + 2(m+1)\epsilon \right\}.$$  (33)

Proof. We leave it to the reader to verify the feasibility of cycle $C^*$. To prove the optimality of $C^*$, we show that its cycle time indeed achieves the lower bound - the expression on the right-hand side of inequation (28).

We analyze the cycle execution. We differ between two possible scenarios. First, suppose that there exists at least one $w_k > 0$, $k \in I_\theta$ (i.e., Step 2.2 is run with robot being non-active for $w_k$ time units while awaiting at machine $M_k$ for a part on this machine to finish processing). Then, it is easy to see that

$$T(C^*) = p_k + 2\epsilon + \theta.$$  (34)

Now, suppose there is no positive $w_k$ (the robot is never forced to spend idle time waiting at any machine $M_k$, $k \in T_\theta$). We then straightforwardly derive:

$$T(C^*) = \sum_{k \in I_\theta} p_k + \sum_{i=1}^{\lceil T_\theta \rceil} \theta + (m+1)\delta + 2(m+1)\epsilon.$$  (35)

Combining (34) and (35) together, we obtain (33). By Lemma 9, the optimality of $C^*$ follows. □

The result of Lemma 9 can be extended onto the general case of $RF^2_m || C_t$ problem (i.e., when the search for a cyclic optimal solution is not restricted to the class of 1–unit cycles only) if we pose an assumption on the values of part processing times.
Lemma 10 For $RF^2_m || C_t$ problem with $\max\{\theta, \theta_t\} \leq \delta$ and $\max_{i=1}^m p_i \geq \delta$, the following lower bound on the per-unit cycle time ($\frac{T(C)}{q}$) of any $q$–unit cycle $C$ holds true:

$$\frac{T(C)}{q} \geq \max \left\{ \max_{1 \leq k \leq m} p_k + \theta + 2\epsilon; m\theta + (m + 1)\delta + 2(m + 1)\epsilon \right\}. \quad (36)$$

Proof. The proof is given in Appendix B. □

By its nature, the above result is similar to the analogous results obtained by Brauner et al. [7] for a robotic cell with machine buffers and by Geismar et al. [14] for a constant travel time cell (the robot travel time between any two machines is a constant $\delta$, see Dawande et al. [12]).

Corollary 2 For $RF^2_m || C_t$ problem with $\max\{\theta, \theta_t\} \leq \delta$ and $\max_{i=1}^m p_i \geq \delta$, Algorithm OPT1 delivers an optimal solution.

8 Single Gripper Robotic Cells Having An Output Buffer At Each Machine

In this Section, we look at the model which is closely related to the dual-gripper problem $RF^2_m || C_t$ we have studied so far. Namely, we discuss a robotic cell served by a single-gripper robot but allowing for temporary storage of parts by means of machine output buffers. A one-unit capacity buffer at a machine can be viewed as an alternative to the additional robot’s gripper. The aim of this Section is to investigate the model with machine buffers as well as to compare the latter against the dual-gripper robotic cell. We start with the formal definition of the problem.

In what follows, we consider the $RF^1_m | q = 1, b = 1 | C_t$ problem of minimizing the per unit cycle time ($C_t$) in an $m$–machine single gripper robotic flow shop ($RF^1_m$) producing identical parts using 1–unit cycles ($q = 1$), where each machine has an output buffer of one-unit capacity ($b = 1$). The flow-shop processing requirements as well as the cell configuration are the same as in $RF^2_m | q = 1 | C_t$ model (see Section 3). Yet, contrary to the dual-gripper model, in $RF^1_m | q = 1, b = 1 | C_t$ problem, the robot has only one gripper, and thus, can be in possession of at most one part. Furthermore, each machine $M_i$, $i = 1, 2, \ldots, m$, is equipped with a one-unit capacity output buffer $B_i$ (i.e., a buffer may accommodate at most one part). The pair $(M_i, B_i)$ is called the production unit $Z_i$.

Evidently, the main purpose of an output buffer at a machine is to allow for temporary storage of a part that has been processed on the machine, so as to vacate this machine and make it possible for the machine to “take” another part. From the optimization point of view, we make the following assumptions on the use of a production unit $(M_i, B_i)$:

(i) Buffer $B_i$ is used to accommodate a part $(P_j)$ that has finished its processing on $M_i$ if and only if
machine $M_i$ is scheduled to be loaded with the next part ($P_{j+1}$) before part $P_j$ leaves the production unit ($M_i, B_i$).

**(ii)** The robot is not allowed to load part $P_{j+1}$ on $M_i$ until part $P_j$ is moved securely to the buffer. This is done for the safety reason to avoid any collusion of the robot with the local material handling device. We observe this in a Dallas based company.

Observe, that assumption (i) makes the use of a buffer to be cycle-dependent. That is, the decision on whether a part ($P_j$), that has finished its processing on a machine $M_i$, has to be transferred to buffer $B_i$ depends on the next robot operation scheduled for unit $Z_i$. Namely, if the next operation is “load machine $M_i$” then part $P_j$ goes to $B_i$ immediately upon completion its processing on $M_i$ (and so, machine $M_i$ is vacant and ready to take the next part ($P_{j+1}$) as soon as possible). Otherwise, when there is no necessity to use a buffer, part $P_j$ sits on machine $M_i$ until the robot comes to $M_i$ and unloads it. In such a way, we optimize the use of machine-buffer resources (the time to unload a part from a machine and to move it to this machine’s buffer is spent only when it is necessary for a cycle to be executable).

Below we define the problem time parameters:

- $p_i$: the processing time of a part on machine $M_i$, $i = 1, 2, \ldots, m$.
- $\epsilon$: the time of load/unload operation executed by the robot, which is the time taken by the robot to pick up/drop off a part at $I/O$; also the time taken by the robot to perform load /unload operation at any production unit $Z_i$, $i = 1, 2, \ldots, m$ (i.e., load/unload machine $M_i$ or unload buffer $B_i$, whatever the robot is scheduled to do).
- $\delta$: the time taken by a rotational robot movement when traveling between two consecutive production units $Z_{j-1}$ and $Z_j$, $1 \leq j \leq m + 1$. The travel times are additive for nonconsecutive machines/buffers as described in Section 3.
- $\mu$: the time taken by the robot to travel from a machine $M_i$ (after it loads this machine) to this machine’s buffer $B_i$.
- $\alpha$: the total time taken by the local handling device to unload a finished part from a machine $M_i$ and transfer it to this machine’s buffer $B_i$.

We conduct our analysis of the problem under the following assumption:

$$\alpha = \mu + \epsilon. \quad (37)$$
We ought to say here that our choice of $\alpha$ had initially came from the practical consideration (as being observed in real-life robotic cells). It turns out that it also slightly simplifies the mathematical analysis of the problem. Nevertheless, we remark that all results obtained below for $RF_m^1 \mid q = 1, b = 1 \mid C_t$ problem under assumption (37) can be easily extended for the case of general $\alpha$.

We further remark that Brauner et al. [7] consider the problem $RF_m^1 \mid q = 1, b = 1 \mid C_t$ with slightly different set of assumptions. Namely, the Brauner’s model is a special case of our model, where $\mu = \alpha = 0$. Moreover, we note that our model can be easily converted to the model, where a part always goes to a machine’s buffer after its processing on the machine is completed (and so, the use of buffers is uniform). We obtain the latter model by mere setting $p_t^{new} = p_t + \alpha$ and $\alpha^{new} = 0$.

We now discuss the construction and notational representation of 1–unit cycles for $RF_m^1 \mid q = 1, b = 1 \mid C_t$ problem. We denote by $M_k^L \ (M_k^U)$ and $B_k^L \ (B_k^U)$ the operations “load $M_k$” (“unload $M_k$”) and “load $B_k$” (“unload $B_k$”), respectively. We write $M_0^U = I$ and $M_{m+1}^L = O$ to notate the operations “pick up a part from Input” and “drop a part at Output”, respectively. Furthermore, we write $Z_k^L$ and $Z_k^U$ to refer to operations “load unit $Z_k$” and “unload unit $Z_k$”, respectively. Here, $Z_k$ may stand for either $M_k$ or $B_k$.

The sequence of robot activities, “Unload a part ($P_j$) from unit $Z_k$ — Go to machine $M_{k+1}$ — Load part $P_j$ on machine $M_{k+1}$,” is called activity $A_k$. We denote activity $A_k$ by

$$A_k = (Z_k^U - \circ - M_{k+1}^L),$$

(38)

where $Z_k^U \in \{M_k^U, B_k^U\}$ and $\circ \in \{0, [M_{k+1}^U, B_{k+1}^L]\}$. Here, the term “$\circ$” is used to specify whether or not machine $M_{k+1}$ had to be earlier served by the local material handling device that would unload a part previously processed on $M_{k+1}$ and move it to buffer $B_{k+1}$. In what follows, we will exploit the representation (38) in both the general form: $A_k = (Z_k^U - \circ - M_{k+1}^L)$, and an explicit form where $Z_k^U$ and “$\circ$”-term are explicitly specified.

To get the feeling of the situation, consider for example the following two 1–unit cycles for $RF_m^1 \mid q = 1, b = 1 \mid C_t$ problem with $m = 2$:

**Cycle** $C_1 : (I - [M_1^U, B_1^U] - M_1^L), (B_1^U - M_1^L), (M_2^U - O)$. Here, the robot’s activities are as follows: the robot picks a part ($P_j$) from Input ($\epsilon$); goes to machine $M_1$ ($\delta$); if necessary, waits for a part ($P_{j-1}$) on $M_1$ to be completed and transferred to buffer $B_1$ (wait time $w_1$); loads part $P_j$ on $M_1$ ($\epsilon$); moves to buffer $B_1$ ($\mu$); unloads part $P_{j-1}$ from $B_1$ ($\epsilon$); goes to machine $M_2$ ($\delta$); loads part $P_{j-1}$ on $M_2$ ($\epsilon$); waits for $p_2$ time units for part $P_{j-1}$ on $M_2$ to be processed; unloads part $P_{j-1}$
from $M_2$ ($\varepsilon$); goes to $I/O$–station ($\delta$); drops part $P_{j-1}$ onto Output ($\varepsilon$). Under assumption (37), the cycle time of cycle $C_1$ is

$$T(C_1) = \max \{ p_1 + \mu + 2\varepsilon; p_2 + \mu + 3\delta + 6\varepsilon \}.$$  

Cycle $C_2 : (I - [M_1^U, B_1^I] - M_1^L), (M_2^U - O), (B_2^I - M_2^L)$. Here, the robot picks a part ($P_j$) from Input ($\varepsilon$); goes to machine $M_1$ ($\delta$); waits, if necessary, for a part ($P_{j-1}$) on $M_1$ to be completed and transferred to buffer $B_1$ ($w_1$); loads part $P_j$ on $M_1$ ($\varepsilon$); goes to machine $M_2$ ($\delta$); waits, if necessary, for a part ($P_{j-2}$) on $M_2$ to be completed and transferred to buffer $B_1$ ($w_2$); unloads part $P_{j-2}$ from $M_2$ ($\delta$); goes to $I/O$–station ($\delta$); drops part $P_{j-2}$ onto Output ($\varepsilon$); goes to buffer $B_1$ ($\delta$); unloads part $P_{j-1}$ from $B_1$ ($\varepsilon$); goes to machine $M_2$ ($\delta$); loads part $P_{j-1}$ on $M_2$ ($\varepsilon$); goes to $I/O$–station ($\delta$). Under assumption (37), the cycle time of cycle $C_1$ is

$$T(C_2) = \max \{ p_1 + \mu + 2\varepsilon; p_2 + 3\delta + 4\varepsilon; 6\delta + 6\varepsilon \}.$$  

In any 1–unit cycle, each of the operations “load machine $M_k$”, $k = 1, 2, \ldots, m + 1$, is performed exactly once. Furthermore, for any $k, k = 0, 1, \ldots, m$, the robot operation “unload a part ($P_j$) from unit $Z_k$” is always immediately followed by “load part $P_j$ onto machine $M_{k+1}$”, i.e., after picking up part $P_j$ from unit $Z_k$ the robot has no choice for its next operation but to go to machine $M_{k+1}$ and load the latter with part $P_j$. We thus have

**Property 7** For $RF^1_m | q = 1, b = 1 | C_t$ problem, any feasible 1–unit cycle corresponds to the unique sequence of activities $A_k = (Z_k^U - o - M_{k+1}^L), k = 0, 1, \ldots, m$ (under the obvious proviso that the latter sequence is treated in a cyclic manner).

The next result establishes the reverse relationship between 1–unit cycles and activities $A_k$.

**Property 8** For $RF^1_m | q = 1, b = 1 | C_t$ problem, any permutation of activities $A_k, k = 0, 1, \ldots, m,$ written in form (38) where $Z_k^U \in \{ M_k^U, B_k^U \}$ are specified (i.e., an activity $A_k$ is in the form of either $A_k = (M_k^U - o - M_{k+1}^L)$ or $A_k = (B_k^U - o - M_{k+1}^L)$), defines a unique 1–unit cycle.

Proof. We first observe that, by specifying $Z_k^U \in \{ M_k^U, B_k^U \}$ in activity $A_k$, we determine how the buffer $B_k$ is used. In particular, if in $A_k$ we have $Z_k^U = B_k^U$, i.e., $A_k = (B_k^U - o - M_{k+1}^L)$, then “$o$”-term of activity $A_{k-1}$ must be $[M_k^U, B_k^U]$, i.e. $A_{k-1} = (Z_{k-1}^U - [M_k^U, B_k^U] - M_{k+1}^L)$. Otherwise, if $Z_k^U = M_k^U$ (so that $A_k = (M_k^U - o - M_{k+1}^L)$) then $A_{k-1} = (Z_{k-1}^U - M_k^L)$ (“$o$”-term is empty). Thus, by specifying the $Z_k^U$–entries of $A_k$, we uniquely identify the “$o$”-terms of all activities, and therefore, both the robot
activities and the operations within production units are well-defined. Hence, a permutation of activities \(A_k, k = 0, 1, \ldots, m\), written in form (38) where \(Z^U_k \in \{M^U_k, B^U_k\}\) are specified, completely defines the unique cycle. We leave it to the reader to verify that such a cycle is always feasible. \(\Box\)

The following lemma gives a total number of 1–unit cycles for an \(m\)–machine single-gripper robotic cell with unit-capacity machine buffers.

**Lemma 11** For \(RF^1_m | q = 1, b = 1 | C_t\) problem, the number of 1–unit cycles is equal to \(m! \times 2^m\).

Proof. Clearly, any 1–unit robot cycle for \(RF^1_m | q = 1, b = 1 | C_t\) problem, if represented by a permutation of activities \(A_k, k = 0, 1, \ldots, m\), admits \((m + 1)\) different representations, depending on which activity is chosen as the starting activity. To achieve the uniqueness of cycle representation, let us demand that the cycle always starts with \(A_0 = (I \circ - M^L_{1})\). With this assumption, by Property 7, a 1–unit cycle admits the one and only representation. The latter is defined by permutation of the remaining \(m\) activities \(A_k, k = 1, 2, \ldots, m\), as well as by the \(Z^U_k\)–entries of activities \(A_k\). There are \(m!\) different permutations of activities \(A_k, k = 1, 2, \ldots, m\), written in form \((Z^U_k \circ - M^L_{k+1})\). As \(Z^U_k\) may stand for either \(M^U_k\) or \(B^U_k\), any permutation of activities \(A_k = (Z^U_k \circ - M^L_{k+1}), k = 1, 2, \ldots, m\), expands into \(2^m\) different permutations of \(A_k\) with explicitly specified \(Z^U_k \in \{M^U_k, B^U_k\}\). By Property 8, any such a permutation of activities \(A_k\) defines a unique 1–unit cycle for \(RF^1_m | q = 1, b = 1 | C_t\) problem. The Lemma follows. \(\Box\)

Let

\[ I_\mu = \{k | p_k \leq \mu, \ 1 \leq k \leq m\}, \quad I_\mu^\prime = I \setminus I_\mu, \quad r = |I_\mu^\prime|. \tag{39} \]

**Lemma 12** For \(RF^1_m | b = 1 | C_t\) problem with

\[ \mu \leq \delta \tag{40} \]

the following lower bound on the per unit cycle time \(\frac{T(C)}{q}\) of any \(q\)–unit cycle \(C\) holds true

\[ \frac{T(C)}{q} \geq \max \left\{ \max_{1 \leq k \leq m} p_k + \mu + 2\epsilon; \sum_{k \in I_\mu} p_k + r\mu + (m + 1)(\delta + 2\epsilon) \right\}. \tag{41} \]

Proof. The proof is similar to the analogous result obtained by Brauner et al. [7]. Let \(C\) be an arbitrary \(q\)–unit cycle for \(RF^1_m | b = 1 | C_t\) problem written in terms of activities \(A_k, k = 0, 1, \ldots, m\). Note, that each activity \(A_k = (Z^U_k \circ - M^L_{k+1}), k = 0, 1, \ldots, m\), occurs in \(C\) exactly \(q\) times. We
first estimate cycle time as the time spent by the robot to execute the cycle. To execute an activity

\[ A_k = (Z_k^U - \circ - M_{k+1}^L) \]

robot needs at least \(2\epsilon + \delta \) time units: operations \( Z_k^U \) and \( M_{k+1}^L \) take \( \epsilon \) time each and the robot travel time from \( Z_k \) to \( M_{k+1} \) is \( \delta \). The time \( \tau \) spent by the robot between two successively executed activities, say \( A_k \) and \( A_l \), where \( A_k = (Z_k^U - \circ - M_{k+1}^L) \) is immediately followed

\[ A_l = (Z_l^U - \circ - M_{l+1}^L) \]

is estimated as follows:

\[
\begin{align*}
\tau & = 0 & \text{if } M_{k+1}^L = O \text{ and } Z_l^U = I; \\
\tau & = p_{k+1} & \text{if } Z_l^U = M_{k+1}^U; \\
\tau & = \mu & \text{if } Z_l^U = B_{k+1}^U; \\
\tau & = \text{robot travel time from } M_{k+1} \text{ to } Z_l & \delta \geq \epsilon.
\end{align*}
\]

We thus derive

\[
T(C) \geq q \times \left( \sum_{i=1}^{m} \min\{p_i, \mu, \delta\} + (m + 1)(\delta + 2\epsilon) \right)
\]

By \(40\), \( \min\{p_i, \mu, \delta\} = \min\{p_i, \mu\} \) for any \(i = 1, 2, \ldots, m\). Hence, we obtain

\[
\frac{T(C)}{q} \geq \sum_{i=1}^{m} \min\{p_i, \mu\} + (m + 1)(\delta + 2\epsilon) = \sum_{k \in \ell,\mu} p_k + \tau \mu + (m + 1)(\delta + 2\epsilon). \tag{42}
\]

We now look at cycle time from the different point of view. Consider any arbitrary machine \( M_k, k \in \{1, 2, \ldots, m\} \). Let \( \tau' \) be the minimum time spent in partial execution of cycle \( C \) between any two successive occurrences of \( M_k^L \) and \( M_k^U \). Furthermore, let \( \tau'' \) denote the minimum time spent in partial cycle execution between the moment an operation \( M_k^U \) starts and the moment the next successive operation \( M_k^L \) finishes. In \( q \)-unit cycle \( C \), each of the operations “load \( M_k \)” and “unload \( M_k \)” is executed \( q \) times, and hence,

\[
T(C) \geq q \times (\tau' + \tau''). \tag{43}
\]

We now estimate the values of \( \tau' \) and \( \tau'' \). Clearly, one must have \( \tau' \geq p_k \). As for \( \tau'' \), we have two possibilities. If the corresponding cycle subsequence is \((M_k^U - B_k^L) - M_k^L\) (i.e., the subsequence that delivers the minimal value for \( \tau'' \)), then we straightforwardly derive:

\[
\tau'' = \alpha + \epsilon = \mu + 2\epsilon.
\]

Otherwise, the corresponding cycle subsequence takes the form of \((M_k^U - \circ - M_{k+1}^L \ldots Z_{k-1}^U - M_k^L)\), and \( \tau'' \) must comprise time to perform load/unload \( M_k \) (2\(\epsilon\)) as well as the robot travel time forth and back from \( M_k \) to some other machine(s) \( (\geq \delta) \), and so, \( \tau'' \geq \delta + 2\epsilon \). Thus, under condition \( (40) \), we always have \( \tau'' \geq \mu + 2\epsilon \). By combining the above estimations for \( \tau' \) and \( \tau'' \) with \( (43) \), we deduce

\[
\frac{T(C)}{q} \geq p_k + \mu + 2\epsilon, \quad k = 1, 2, \ldots, m.
\]

The latter relation taken together with \( (42) \) yields the desired bound \( (41) \). □
Algorithm OPT2

**Input:** The problem instance of \( RF_1^m | q = 1, b = 1 | C_t \) problem. The sets \( I_\mu \) and \( \overline{I_\mu} \) are as defined by (39).

**Output:** A 1–unit cycle \( \bar{C} \).

**Initial setting:** Units \( Z_k, k \in \overline{I_\mu} \), are occupied with a part; units \( Z_k, k \in I_\mu \), are empty. The robot is positioned at \( I/O \)-station; a robot’s gripper is empty.

**Step 1** The robot picks up a part from Input buffer at \( I/O \) hopper.

**Step 2** For \( k \) from 1 to \( m \):

1. **Step 2.1** The robot moves to machine \( M_k \).
2. **Step 2.2** If \( k \in \overline{I_\mu} \) then
   - if necessary, the robot waits for \( w_k \) time units for a part \( (P_i) \) on \( M_k \) to be completed and transferred to its output buffer \( B_k \);
   - the robot loads a part \( (P_{i+1}) \) on \( M_k \) (\( \epsilon \));
   - the robot moves to buffer \( B_k \) (\( \mu \));
   - the robot unloads the part \( (P_i) \) from \( B_k \) (\( \epsilon \)).
3. Otherwise (if \( k \in I_\mu \))
   - the robot loads a part \( (P_i) \) on \( M_k \) (\( \epsilon \));
   - the robot waits for \( p_k \) time units for the part \( (P_i) \) on \( M_k \) to be processed;
   - the robot unloads the part \( (P_i) \) from \( M_k \) (\( \epsilon \)).

**Step 3** The robot moves to \( I/O \) hopper; drops a part onto Output device of \( I/O \) hopper.

**Theorem 3** For \( RF_1^m | b = 1 | C_t \) problem with \( \mu \leq \delta \), Algorithm OPT2 produces an optimal cycle \( \bar{C} \). The cycle time of cycle \( \bar{C} \) is

\[
T(\bar{C}) = \max \left\{ \max_{k \in \overline{I_\mu}} p_k + 2\epsilon + \mu; \sum_{k \in I_\mu} p_k + r\mu + (m + 1)\delta + 2(m + 1)\epsilon \right\}.
\]

**Proof.** The proof of (44) is similar to that of Theorem 2. By Lemma 12, the optimality of \( \bar{C} \) follows. \( \square \)

We observe that \( \theta \) in the problem \( RF_2^2 \) is somewhat analogous to \( \mu \) in the problem \( RF_1^1 \).

**Corollary 3** Under condition \( \max \{\theta, \theta_t, \mu\} \leq \delta \), problems \( RF_2^2 | q = 1 | C_t \) and \( RF_1^1 | b = 1 | C_t \) have the same minimum cycle time if \( \theta = \mu \).

**Proof:** Follows from Corollary 2 and Theorem 3. \( \square \)

**Corollary 4** Under conditions \( \max \{\theta, \theta_t, \mu\} \leq \delta \) and \( p_i \geq \delta \), \( i = 1, \ldots, m \), problems \( RF_2^2 | C_t \) and \( RF_1^1 | b = 1 | C_t \) have the same minimum cycle time if \( \theta = \mu \).
Proof: Follows from Theorems 2 and 3. □

We wrap up our study on the relationship between problems RF\(_m^2|q = 1|C_t\) and RF\(_m^1|q = 1, b = 1|C_t\) with a simple result that establishes the relative equivalence of these two problems.

In what follows, the robot traveling between any two machines is called an empty move if the robot travels empty, and a loaded move, otherwise. Furthermore, for RF\(_m^1|b = 1|C_t\) problem, a production unit \(Z_i\) is said to be fully-loaded if both \(M_i\) and \(B_i\) are each loaded with a part.

**Definition 6** A cycle \(C\) for RF\(_m^2|q = 1|C_t\) problem is called simple if in \(C\) the robot never carries two parts in its loaded move.

**Definition 7** A cycle \(C''\) for RF\(_m^1|q = 1, b = 1|C_t\) problem is called simple if in \(C''\) the robot never makes an empty move from a fully-loaded unit.

Let \(\Omega'\) and \(\Omega''\) denote the sets of all simple 1–unit cycles for RF\(_m^2|q = 1|C_t\) and RF\(_m^1|q = 1, b = 1|C_t\) problems, respectively. We remark that a trivial example of a simple cycle in either \(\Omega'\) or \(\Omega''\) is a cycle for RF\(_m^1|q = 1|C_t\) problem.

**Lemma 13** Let \(\theta_t \leq \delta\) and \(\theta = \mu\). Then for any cycle \(C' \in \Omega'\) there exists a cycle \(C'' \in \Omega''\) such that

\[
T(C') = T(C''),
\]

and vice versa.

Proof. The cycles \(C' \in \Omega'\) and \(C'' \in \Omega''\) that satisfy (45) are obtained from each other as follows. Both cycles are defined by the very same schedule of robot operations with one and only one exemption. Namely, the sequence of operations “unload machine \(M_k\) - switch grippers - load machine \(M_k\)” in \(C'\) translates into “load machine \(M_k\) - go to buffer \(B_k\) - unload buffer \(B_k\)” for cycle \(C''\). We leave it for the reader to verify that, for given problem data (i.e., for given values of \(p_i, i = 1, 2, \ldots, m, \delta, \epsilon, \) and \(\theta = \mu\)), both cycles deliver the same cycle time. □

Evidently, the above result shows the relative equivalency of RF\(_m^1|q = 1, b = 1|C_t\) and RF\(_m^2|q = 1|C_t\) problems that arises when the utilization of machine buffers (for RF\(_m^1|q = 1, b = 1|C_t\)) and the additional robot’s gripper (for RF\(_m^2|q = 1|C_t\)) is limited to mere part swapping at machines. To give the feeling of the situation, below we give an example of two 1–unit cycles \(C' \in \Omega'\) and \(C'' \in \Omega''\) that comply with the above Lemma:

**Cycle** \(C' : I - M_1^L - M_1^U - M_2^L - M_3^U - M_4^U - M_4^L - O - M_2^U - M_3^L\);
Cycle $C'' : I - M_1^U - M_1^U - M_2^U - M_3^U - M_4^U - B_4^U - O - M_2^U - M_3^U$.

Finally, we look at the possible impact of multi-unit cycles on the cycle time for both $RF_m^2 | C_t$ and $RF_m^1 | b = 1 | C_t$ problems.

Consider the following two-unit cycle, $\hat{C}$, for $RF_m^2 | q = 2 | C_t$ problem:

$$
\hat{C} = (0, m^+, 0^+, 1^-, 0^+, 1^-, \ldots, k - 1^+, k^-, k - 1^+, k^-, \ldots, m - 1^+, m^-, m - 1^+, m^-, m^+) .
$$

The cycle time of cycle $\hat{C}$ is

$$
T(\hat{C}) = (m + 1)\delta + 4(m + 1)\epsilon + (m + 1)\theta + 2\sum_{i=1}^{m} p_i .
$$

Now consider the following problem data:

$$
\theta_i = \theta = \mu = \epsilon = p_i = x, \quad i = 1, 2, \ldots, m; \quad \delta = X .
$$

Let $x$ be a very small positive number, and let $X$ be sufficiently large: $0 < x << X$. For this problem instance, we derive:

$$
T(\hat{C}) = (m + 1) X + (7m + 5) x .
$$

Let $OPT(X)$ denote the optimal per-unit cycle time for problem $X$, for a problem instance defined by (46). For problem $RF_m^2 | q = 2 | C_t$, cycle $\hat{C}$ delivers an optimal solution for the problem data (46), i.e. we have

$$
OPT(RF_m^2 | q = 2 | C_t) = \frac{T(\hat{C})}{2} = \frac{1}{2} ((m + 1) X + (7m + 5) x) .
$$

Consider now $RF_m^1 | b = 1 | C_t$ and $RF_m^2 | q = 1 | C_t$ problems. By Theorems 2 and 3, we have

$$
OPT(RF_m^2 | q = 1 | C_t) = OPT(RF_m^1 | b = 1 | C_t) = (m + 1) X + (3m + 2) x .
$$

As $x \to 0$ and $X \to \infty$, we thus obtain,

$$
\frac{OPT(RF_m^2 | q = 1 | C_t)}{OPT(RF_m^2 | q = 2 | C_t)} = \frac{OPT(RF_m^1 | b = 1 | C_t)}{OPT(RF_m^1 | q = 2 | C_t)} = \frac{2((m + 1) X + (3m + 2) x)}{(m + 1) X + (7m + 5) x} \to 2 .
$$

As a consequence of the above example taken along with Corollary 3, we state the following

**Lemma 14** Under conditions $\max\{\theta, \theta_i\} \leq \delta$ and $\theta = \mu$, the long-run average throughput rate of a dual-gripper cell ($RF_m^m | C_t$) is equal to or greater than that of the single gripper cell having an output buffer at each machine ($RF_m^1 | b = 1 | C_t$).
We close this Section with the summary on the results obtained above and brief concluding remarks. The problem $RF_{m}^{1} | b = 1 | C_t$ can be viewed as a counterpart of the dual-gripper problem $RF_{m}^{2} || C_t$ in sense that an output buffer at the machine plays the same role as an extra robot gripper - they both allow for temporary storage of a part. While the use of output machine buffers offers certain time-flexibility in part storage (as we are less constrained in how long a part can reside at the buffer), the use of an additional robot gripper makes the storage more time-efficient (as we can economize on the robot travel time). Obviously, either model has its benefits, and the assessment of their comparative efficiency may generally depend on the cell parameters as well as the problem data. Above, we suggested the notational framework to represent 1−unit cycles for a robotic cell with machine output buffers, and described the complete family of 1−unit cycles for $RF_{m}^{1} | b = 1 | C_t$ problem. We have shown that a special case of $RF_{m}^{1} | b = 1 | C_t$ is efficiently solvable, with the optimal solution being found in the class of 1−unit cycles. We have compared the performance of a single-gripper cell with output buffers with that of a dual-gripper cell. Furthermore, we provided an example to quantify a possible gain in cell’s productivity for $RF_{m}^{2} || C_t$ problem if compared to the problem $RF_{m}^{1} | b = 1 | C_t$, which can approach 100% in the best, however extreme, case.

Remark 5 So far, we have studied cells, in which all the machines $M_i$, $i = 0, 1, \ldots , m$, are placed equidistantly around a circle, the travel time between two adjacent machines $M_i$ and $M_{i+1}$, $i = 0, 1, \ldots , m$, is constant $\delta$, and the travel times are additive for nonconsecutive machines. Such cells are called additive travel time cells. For certain cells, additive travel-times are not appropriate. Dawande et al. [12] discuss a type of cells (called constant travel time cells) for which the robot travel-time between any pair of machines is a constant $\delta$, i.e., $\ell(M_i, M_j) = \delta$, $0 \leq i, j \leq m + 1$. The latter model arises in compact cells where the robot moves with varying acceleration and deceleration. All the results obtained in this paper for additive travel times will also hold for constant travel time cells.

9 Conclusions

The paper considers the problem of cyclic production of identical parts in robotic cells served by a dual-gripper robot. In this work, we develop new notational and modelling framework for cyclic production in a dual-gripper robotic cell, and mainly focus our attention to derive all active 1−unit cycles. Since 1−unit robot move cycles are the easiest to implement and control, our work has direct implications for the practical operation of robotic cells. The results obtained show that the increase in number of machines in a cell leads to the explosive growth in the combinatorial possibilities in search for an optimal robot moves strategy. We also provide the algorithmic approach to describe the complete family of all
such cycles. Moreover, for a special case of sufficiently small gripper switching time in an $m$–machine, we devise a polynomial time algorithm for finding an optimal 1–unit cycle. In practical situations, this problem variant is in frequent occurrence, and possibly, reflects the most common case in real-life robotic systems. We establish the connection between a dual gripper cell and a single gripper cell having a one-unit output buffer at each machine. We also provide some evidence to demonstrate that a dual gripper cell can be more productive than a single gripper cell with output buffers. Apart from the pure theoretical interest, the comparative analysis of the above two robot models presented in this paper is useful for managers of real-life robotic production systems. It may give helpful insights to aid in assessing the overall costs of a robotic cell against possible cell productivity, which is the paramount issue in real-life managerial decisions.

Acknowledgments

The research was supported in part by the Natural Sciences and Engineering Research Council of Canada (Grant number OGP0104900) and the University of Texas at Dallas.

Appendix A: Proof of Lemma 3

If part: Here we prove the sufficiency of (i)-(v) conditions, i.e., we show that any $S$–sequence, which satisfies the above (i)-(v) conditions of the lemma, is feasible. We prove it by contradiction.

Suppose, for a contradiction, that there exists a sequence $S = (s_1, s_2, \ldots, s_{2(m+1)})$, for which the conditions (i)-(v) are met, but it is not feasible. By virtue of Lemma 2, it implies that, for this $S$–sequence and for some $k \in \{0, 1, \ldots, m\}$, there exists $S_k$-sequence defined by (4), for which the requirement (6)-(7) posed by Lemma 2 is violated. That is, in such $S_k$-sequence we may identify some $r \in \{0, 1, \ldots, m\} \backslash \{k\}$, so that one of the following inequalities holds valid

$$1 = p_{S_k}(k^-) < p_{S_k}(r^-) < \min\{p_{S_k}(k^+), p_{S_k}(r^+)\} - 1; \quad (47)$$

$$1 = p_{S_k}(k^-) < p_{S_k}(r^+) < p_{S_k}(r^-) < p_{S_k}(k^+) - 1. \quad (48)$$

We will consider each of these possibilities separately. In each case we will follow the same scheme. Assuming that the corresponding inequality ((47) or (48)) holds true for $S_k$ while all (i)-(v) conditions of Lemma are met for $S$, we will aim to identify a position of $s_1$ in $S_k$. Naturally, the failure to succeed in such after having exhausted all possibilities, i.e., proving that $p_{S_k}(s_1) \notin \{1, 2, \ldots, 2(m+1)\}$, will imply that the initial assumption of infeasibility of $S$ has been wrongful.

We start with the situation when (47) takes place. Obviously, $k^- \neq s_1$ ($p_{S_k}(k^-) \neq p_{S_k}(s_1)$) and also $p_{S_k}(s_1) \leq \max\{p_{S_k}(k^+), p_{S_k}(r^+)\}$, since otherwise the condition (i) is violated for $S$ with $x = k$ and $y = r$. 

40
Next, if $p_{S_k}(k^-) < p_{S_k}(s_1) \leq p_{S_k}(r^-)$, we have the violation of condition (ii) for $S$ with $x = r$ and $y = k$. Furthermore, if $p_{S_k}(r^-) < p_{S_k}(s_1) < \min\{p_{S_k}(k^+), p_{S_k}(r^+)\}$ we have that $1 < \min\{p_{S}(k^+), p_{S}(r^+)\} < \max\{p_{S}(k^+), p_{S}(r^+)\} < \min\{p_{S}(k^-), p_{S}(r^-)\}$, and so the condition (v) (equality (10)) is at error for $S$ with $x, y \in \{k, r\}$. If $p_{S_k}(s_1) = \min\{p_{S_k}(k^+), p_{S_k}(r^+)\}$, then $p_{S_k}(r^-) < p_{S_k}(s_1) - 1$, and so in $S$ we have $1 = \min\{p_{S_k}(k^+), p_{S_k}(r^+)\} < \max\{p_{S}(k^+), p_{S}(r^+)\} < \min\{p_{S}(k^-), p_{S}(r^-)\} < 2(m + 1)$, i.e. the condition (v) (equality (11)) is at error. Hence, the only option left for $p_{S_k}(s_1)$ is $\min\{p_{S_k}(k^+), p_{S_k}(r^+)\} < p_{S_k}(s_1) \leq \max\{p_{S_k}(k^+), p_{S_k}(r^+)\}$. But in that case we face the violation of condition (iii) for $S$ with $x, y \in \{k, r\}$, such that $p_{S_k}(x) = \min\{p_{S_k}(k^+), p_{S_k}(r^+)\}$ and $p_{S_k}(y) = \max\{p_{S_k}(k^+), p_{S_k}(r^+)\}$. Thus, we come to the desired conclusion: $p_{S_k}(s_1) \notin \{1, 2, \ldots , 2(m + 1)\}$.

We then move to the situation when (48) takes place. Again, we look at all possibilities to position $s_1$ in $S_k$. Obviously, $p_{S_k}(k^-) < p_{S_k}(s_1) \leq p_{S_k}(k^+)$, since otherwise the condition (iv) is violated for $S$ with $x = k$ and $y = r$ (equality (9)) is at error. Next, if $p_{S_k}(k^-) < p_{S_k}(s_1) \leq p_{S_k}(r^+)$ we have $p_S(s_1) \leq p_S(r^-) < p_S(r^-) < p_S(k^-) - 1 < p_S(k^-) < p_S(k^-)$. And so, we encounter the situation outlined in condition (v) with $x, y \in \{k, r\}$ and (12) being satisfied, while equality (13) being violated, thus, condition (v) does not hold true. Furthermore, $p_{S_k}(r^+) < p_{S_k}(s_1) \leq p_{S_k}(r^-)$ would lead to the straightforward violation of condition (v) (equality (8)) for $S$ with $x = r$ and $y = k$. And, therefore, we are left with the last remaining option for positioning $s_1$ in $S_k$: $p_{S_k}(r^-) < p_{S_k}(s_1) \leq p_{S_k}(k^+)$ with $p_{S_k}(r^-) < p_{S_k}(k^+) - 1$. But here we will evidently have the violation of condition (v) for $S$ with $x, y \in \{k, r\}$, namely equation (10) does not hold valid if $p_{S_k}(s_1) < p_{S_k}(k^+)$, whereas $p_{S_k}(s_1) = p_{S_k}(k^+)$ implies violation of (11). Again, we end up with the conclusion that $p_{S_k}(s_1) \notin \{1, 2, \ldots , 2(m + 1)\}$.

Only If part: We now prove the necessity of (i)-(v) conditions. We show that violation of any one of (i)-(v) conditions of the lemma implies that a sequence $S$ is not feasible, i.e., the feasibility requirement of Lemma 2 is violated for some $S_k$, which may be constructed from $S$. We will consider (i)-(v) conditions, one by one. In each case, we identify sequence $S_k$ for which the feasibility requirement of Lemma 2 is violated. Suppose that, for some $S$, condition (i) is not valid. Then the feasibility requirement of Lemma 2 is violated for $S_k$ with $k = x$. Next, suppose condition (ii) is not satisfied, for some $S$. Then the feasibility requirement of Lemma 2 is violated for $S_k$ with $k = y$. Furthermore, if, for some $S$, condition (iii) does not hold true, the feasibility requirement of Lemma 2 does not hold true as well for $S_k$ with $k \in \{x, y\}$ such that $p_S(k^-) = \min\{p_S(x^-), p_S(y^-)\}$. The violation of condition (iv) for some $S$ implies the violation of the feasibility requirement in Lemma 2 for $S_k$ with $k = x$ if (9) is not valid, and for $S_k$ with $k = y$ if (8) is not satisfied. Finally, for condition (v), the violation either of equations (10)-(11) implies the violation of Lemma 2 requirement for $S_k$ with $k \in \{x, y\}$ such that $p_S(k^-) = \min\{p_S(x^-), p_S(y^-)\}$. 
In turn, the violation of equation (13), provided that (12) takes place, suggests the violation of the feasibility requirement of Lemma 2 for $S_k$ with $k \in \{x, y\}$ such that $p_S(k^-) = \max\{p_S(x^-), p_S(y^-)\}$. This proves the lemma. □

**Appendix B: Proof of Lemma 10**

The proof follows along the same lines as that of Lemma 9. Let $C$ be an arbitrary $q$–unit robot cycle, represented by a sequence $S = (s_1, s_2, \ldots, s_{2q(m+1)})$ of “load a machine” and “unload a machine” operations. By Property 1, each of $[k^-] / [k^+]$–entries, $k \in \{0, 1, \ldots, m\}$, occurs in $S$ exactly $q$ times.

Consider any particular machine $M_k$, $k = 1, 2, \ldots, m$. For cycle $C$, let $\tau$ denote the minimum time taken by a partial cycle execution between any two successive starting moments of “load $M_k$” operation ($[k-1]^+$). We then have

$$T(C) \geq q \times \tau. \quad (49)$$

Note, that in case of $q = 1$, the above relation becomes the equality. Evidently, between any two “load $M_k$” operations there should be one “unload $M_k$” operation ($M_k$ must be vacated from the previous part to be loaded with the next one). Thus, $\tau$ necessarily contains part processing time ($p_k$), time to load and unload $M_k$ ($2\epsilon$), as well as the time spent between “unload $M_k$” and the next “load $M_k$”. By the same argument as in the proof of Lemma 9, the latter either equals $\theta$ or comprises the robot travel time ($\geq \delta$) forth/back from $M_k$ to some other machine. We, therefore, have $\tau \geq p_k + \theta + 2\epsilon$. By (49), we derive

$$\frac{T(C)}{q} \geq p_k + \theta + 2\epsilon, \quad k = 1, 2, \ldots, m. \quad (50)$$

Furthermore, for sequence $S = (s_1, s_2, \ldots, s_{2q(m+1)})$, let contribution $\Delta_{s_j}$ to the cycle time $T(C)$ for a particular $s_j$, $j = 1, 2, \ldots, 2q(m+1)$, be as defined in the proof of Lemma 9. We then have

$$T(C) = \sum_{j=1}^{2q(m+1)} (\Delta_{s_j} + \epsilon) = 2q(m+1)\epsilon + \sum_{j=1}^{2q(m+1)} \Delta_{s_j}. \quad (51)$$

We will now estimate the contributions $\Delta_{s_j}$. Let $I = \{1, 2, \ldots, 2q(m+1)\}$. Repeating the observations made in the proof of Lemma 9, we have the following:

$$\Delta_{s_j} \geq \min\{p_k, \delta\} = \delta, \quad \text{for any } s_j = [k^-], \quad k \in \{1, 2, \ldots, m\},$$

$$\Delta_{s_j} \geq \min\{\theta, \delta\} = \theta, \quad \text{for any } s_j = [k^+], \quad k \in \{0, 1, \ldots, m-1\}.$$  

Hence, we obtain

$$\sum_{k=1}^{m} \sum_{j \in I, \ s_j = [k^-]} \Delta_{s_j} \geq qm\delta; \quad (52)$$
\[
\sum_{k=1}^{m-1} \sum_{j \in I, \ s_j=[k]^+} \Delta_{s_j} \geq q(m-1)\theta. \tag{53}
\]

The latter relations include \(\Delta_{s_j}\) for all \(s_j\) but \([m]^+, \ [0]^-, \text{ and } [0]^+\) (we remark that we deliberately excluded \([0]^+\) from (53) as we will exploit it in our further estimations of \(\Delta_{s_j}\)-contributions for \(s_j \in \{[m]^+, \ [0]^-, \ [0]^+\}\)). By (50)-(53), to prove the theorem it now suffices to show that
\[
\sum_{j \in I, \ s_j\in\{[m]^+, \ [0]^-, \ [0]^+\}} \Delta_{s_j} \geq q(\delta + \theta). \tag{54}
\]

Thus, we aim to prove (54). We start with introducing some useful terms and notation. Operation \([0]^-\) is said to have the rank of 2 \((\text{rank}([0]^-) = 2)\) if upon its completion the robot holds two parts both intended for machine \(M_1\), otherwise the rank of operation \([0]^-\) is set to 1 (the robot holds only one part that goes to \(M_1\)). Furthermore, operation \([0]^+\) is said to have the rank of 2 \((\text{rank}([0]^+) = 2)\) if the robot comes to \(M_1\) to perform this operation holding two parts both intended for machine \(M_1\), otherwise the rank of operation \([0]^+\) is set to 1. Given sequence \(S\), we further define the following sets of \(j\)-indices for operations \(s_j = [0]^-\) and \(s_j = [0]^+\):
\[
J' = \{j \mid s_j = [0]^-, \ \text{rank}(s_j) = 1, \ j \in I\} \quad \text{and} \quad J'' = \{j \mid s_j = [0]^-, \ \text{rank}(s_j) = 2, \ j \in I\} ; \\
J = \{j \mid s_j = [0]^+, \ \text{rank}(s_j) = 1, \ j \in I\} \quad \text{and} \quad \overline{J} = \{j \mid s_j = [0]^+, \ \text{rank}(s_j) = 2, \ j \in I\} .
\]

By the above definition of the term “\(\text{rank}\)”, we have the following relations for the sets introduced:
\[
J' \cap J'' = \emptyset, \quad J \cap \overline{J} = \emptyset, \quad |J'| + |J''| = |J| + |\overline{J}| = q. \tag{55}
\]

Furthermore, it is easy to see that for each \(s_j = [0]^-\) with \(j \in J''\) one must have \(s_{j+1} = [0]^+\) with \((j+1) \in \overline{J}\), and hence, \(\overline{J} = \{j + 1 \mid j \in J''\}\), which in turn yields
\[
|J''| = |\overline{J}|. \tag{56}
\]

For \(s_j = [0]^-\), let set \(\Omega_{s_j}\) comprises \(s_j\) and all immediately preceding it \([m]^+\)-entries of \(S\), if any. Namely
\[
\Omega_{s_j} = \begin{cases} 
\{s_j\} & \text{if } s_j = [0]^-, \ s_{j-1} \neq [m]^+; \\
\{s_j, s_{j-1}\} & \text{if } s_j = [0]^-, \ s_{j-1} = [m]^+, \ s_{j-2} \neq [m]^+; \\
\{s_j, s_{j-1}, s_{j-2}\} & \text{if } s_j = [0]^-, \ s_{j-1} = s_{j-2} = [m]^+. 
\end{cases}
\]

For \(s_j = [0]^+\), we then denote by \(\overline{\Delta_{s_j}}\) the sum of \(\Delta_{s_{j'}}\)-contributions for all elements of set \(\Omega_{s_j}\):
\[
\overline{\Delta_{s_j}} = \sum_{s_{j'} \in \Omega_{s_j}} \Delta_{s_{j'}}.
\]

43
Observe that
\[
\sum_{j \in I, s_j \in\{[m]^+, [0]^{-}\}} \Delta_{s_j} \geq \sum_{j \in I, s_j = [0]^{-}} \widetilde{\Delta}_{s_j}. \tag{57}
\]

We now derive the estimations of \(\Delta_{s_j}\) and \(\widetilde{\Delta}_{s_j}\)-contributions for \(s_j = [0]^+\) and \(s_j = [0]^-,\) respectively. We have
\[
\Delta_{s_j} = \delta, \quad \text{for } j \in J, \quad \text{and} \quad \Delta_{s_j} \geq \theta, \quad \text{for } j \in J;
\]
\[
\widetilde{\Delta}_{s_j} \geq \theta, \quad \text{for } j \in J'', \quad \text{and} \quad \widetilde{\Delta}_{s_j} \geq \delta, \quad \text{for } j \in J'.
\]

The above expressions taken together with relations (55)-(57) yield
\[
\sum_{j \in I, s_j \in\{[m]^+, [0]^-, [0]^+\}} \Delta_{s_j} \geq \sum_{j \in I, s_j = [0]^+} \Delta_{s_j} + \sum_{j \in I, s_j = [0]^{-}} \widetilde{\Delta}_{s_j}
\]
\[
= \sum_{j \in J} \Delta_{s_j} + \sum_{j \in J} \Delta_{s_j} + \sum_{j \in J''} \widetilde{\Delta}_{s_j} + \sum_{j \in J'} \widetilde{\Delta}_{s_j} \geq q(\delta + \theta).
\]

And so, (54) holds. \(\Box\)
References


