TR-IIS-07-001

Consistency and Feasibility of Flexible Demand-Supply Constraints

P. H. Tsai and J. W. S. Liu

January 11, 2007 || Technical Report No. TR-IIS-07-001
http://www.iis.sinica.edu.tw/LIB/TechReport/tr2007/tr07.html

Institute of Information Science, Academia Sinica

Technical Report TR-IIS-07-001

Consistency and Feasibility of Flexible Demand-Supply Constraints

P. H. Tsai and J. W. S. Liu[*]

Abstract

A *general schedule specification* (*GSS*) defines constraints on sizes and temporal separation of individual dispatches and upper and lower bounds on the total size of dispatches in specified time intervals. A dispatch may be a dispensing of medications to an individual, a delivery of some fresh produce to a green grocer, the transmission of a multimedia data element to a web surfer, and so on. When given a GSS, the scheduler can choose any schedule that the meets the constraints defined by the specification. The GSS is *consistent* if the constraints defined by it do not conflict with each other and is *feasible* if there is a schedule that meets all constraints. This paper describes conditions and algorithms which the scheduler can use to determine whether the specification is consistent and feasible and to schedule dispatches according to the GSS.

[*] P. H. Tsai is affiliated with Department of Computer Science, National Tsing Hua University, Taiwan. J. W. S. Liu is affiliated with Institute of Information Science, Academia Sinica, Taiwan.

Table of Contents

1 Introduction

Flexibility is a desirable attribute of schedules of many widely disparate activities. Examples include schedules for dispensing medications to individuals, produce deliveries to green grocers and multimedia data transmissions to web surfers. Many individuals today take multiple medications on a long term basis to control chronicle conditions and maintain health. It would be a hardship for even the most disciplined and conscious user to adhere to a rigid, inflexible schedule, day in and day out. Fortunately, directions of most medications allow some variations in sizes of doses and separations between doses. A smart medication dispenser such as the one described in [1] can take advantage of the allowed variations to make its user's medication schedule flexible, conforming to the user's life style and habits whenever possible. Similarly, by using flexible schedules that take full advantage of tolerable variations in amounts and times of deliveries, the schedulers in commodity-flow management and web hosting systems can help the suppliers to better meet the conflicting demands of their customers.

Hereafter, we use the term *dispatches*, rather than application specific terms such as doses and deliveries, except for where it is necessary to be specific. In addition to ranges of sizes and temporal separations of individual dispatches, there may also be bounds on the total size of dispatches over specified time intervals. We refer to these parameters collectively as a *general schedule specification* (*GSS*). It is a subset of the medication schedule specification introduced in [2, 3] to capture directions of multiple medications managed by a smart medication dispenser for a user. The constraints defined by the individual parameters in a GSS are *consistent* if they do not conflict with each other. We call a schedule that meets all the constraints a *feasible schedule* and say that the constraints, and hence the specification, are *feasible* if there is a feasible schedule. A GSS is *consistent and feasible* if the constraints defined by it are consistent and feasible.

In each of the above examples, the scheduler is given a GSS. Before it proceeds to schedule dispatches according to the specification, the scheduler must make sure that the specification is consistent and feasible. This paper describes conditions and algorithms which the scheduler can use for this purpose. The d*emand versus supply test* (*DST*) described here is a more accurate feasibility test than the dosage demand test [1]. As a by-product, the DST test produces a feasible dispatch schedule when it finds the specification feasible. The paper also describes heuristic scheduling algorithms and evaluates their performance along different dimensions.

The problem on feasibility of GSS resembles the schedulability analysis problems treated in real-time systems literatures (e.g. [4-6]). The purpose of schedulability analysis is to determine whether a given set of real-time tasks that share resources has a feasible schedule. The tasks may be infeasible because they demand more resources than what are available. We can also caste our feasibility test problem as that of determining whether the available supply is sufficient to meet the demands. This is where the similarity between the problems ends, however. The problem of scheduling dispatches with size and temporal separation constraints resembles pin-wheel scheduling problems (e.g., [7, 8]) that arise in real-time systems and machine maintenance. An important difference is the requirements defined by bounds on the total size of dispatches in specified intervals, and these bounds are the reason that a consistent GSS may not be feasible. Pin-wheel scheduling is not concerned with such constraints. According to the models used for supply chain assignment and scheduling (e.g., [9]), customers make one-time orders. The goal is to find schedules with minimum weighted total production and distribution cost, subject to some constraints on lead and completion times. It will become evident in later sections that the problem differs significantly from the problem addressed here.

Following this introduction, Section 2 presents a formal definition of GSS and the necessary conditions that a GSS must satisfy to be consistent. Sections 3 and 4 describe the parts of a feasibility test, called demand versus supply test, and discuss the conditions under which the test is accurate. Section 5 describes heuristic algorithms for dispatch scheduling. Section 6 defines figures of merit used to measure the performance of the algorithms and present simulation data on their relative performance. Section 7 is a summary and discusses future work.

2 Constraint Parameters and Their Consistency

Again, our focus is on schedules of activities such as dispensing a medication, delivering a kind of produce, etc. Fig. 1 gives a partial list of parameters provided by a GSS for such a schedule.

• N:	Name of items to be dispatched
• g:	Granularity of dispatches
• τ:	Basic unit of time
• $[d_{min}, d_{max}]$:	Minimum and maximum sizes of dispatches
• $[s_{min}, s_{max}]$:	Minimum and maximum temporal separations between (consecutive) dispatches
• $[T_{min}, T_{max}]$:	Minimum and maximum schedule durations
• (B, R):	Supply rate defined by budget B (maximum total size) over a specified time interval given by replenishment delay R
• (L, P):	Demand rate defined by lower bound L on total size over a specified time interval of length P

Fig. 1 Constraint Parameters

The name N identifies the kind of items to be dispatched and provides physical characteristics and other attributes needed by the scheduler and the application. The granularity g specifies the absolute minimum amount to dispatch each time. For example, a granule of a medication may be a tablet or caplet, or some number of milligrams or cubic centimeters, etc. while a granule of a produce may be a crate or a kilogram. Similarly, the basic unit of time τ is application dependent. It may be 15 minutes, 1/2 hour or an hour for medication administration, but is an hour, a day or a week for schedules of produce and commodity deliveries.

Dispatch size is the number of granules to be dispatched at a time. *Separation* is the length of time interval, measured in terms of number of time units, between any two consecutive dispatches. The specification constrains the size of each dispatch to be in the range $[d_{min},, d_{max}]$ delimited by the *minimum dispatch size* d_{min} and *maximum dispatch size* d_{max}. Similarly, the specification constrains the separations between consecutive dispatches to be within the range $[s_{min}, s_{max}]$ delimited by the *minimum separation* s_{min} and the *maximum separation* s_{max}. All of these parameters are integer valued. This paper focuses on the special case where both the size range $[d_{min},, d_{max}]$ and separation range $[s_{min}, s_{max}]$ are contiguous. In general case, each of these ranges may consist of multiple disjoint ranges. We will discuss the general case in Section 7.

A schedule can be defined in general by a list of 2-tuples (t_i, d_i) of time instants t_i and dispatch size d_i, for $i = 0, 1, \dots, K$. Both t_i and d_i are integers. The list is sorted in ascending order according to t_i. The time origin $t_0 = 0$ is the time of the first dispatch; it is the start of the schedule. The presence of (t_i, d_i) in the list means that a dispatch of d_i granules is scheduled at t_i time units from the start. For example, if the list is the schedule of a medication that comes in tablet form, t_i and d_i give, respectively, the time in minute or hour and the number of tablets of the ith dose according to the schedule. The total number K of dispatches is a function of the duration T of the schedule. T must be in the range delimited by the *minimum duration* T_{min} and the *maximum duration* $T_{max.}$

Fig.1 also gives an upper bound and a lower bound on combinations of dispatch sizes and separations. The *supply rate* (B, R) bounds from above the total dispatch size over an interval of time: No more than B granules of N can be dispatched in any time interval of length R time units. B stands for *budget*, and R stands for *replenishment delay.* Before the first dispatch, the current budget of N is B. When a dispatch of size d is made at time t, d granules of the current budget are consumed, and the d-granule chunk will be replenished at time $t + R$. At the time of any dispatch, the dispatch size can be no more than the current budget. This way of controlling the total size of dispatches over time resembles the way some sporadic servers in real-time systems control the processor bandwidth consumed by aperiodic tasks [10-12]. As examples, the direction of a pain killer whose effect takes 24 hours to dissipate typically includes something like "no more than 6 tablets in 24 hours" to prevent overdose. The supplier of a produce that takes two weeks to mature may impose an upper limit of "at most 100 crates in 14 days" to prevent exhaustion of the supply. These supply rate limits are specified as (6, 24) and (100, 14), respectively, in the GSS for the pain killer and produce.

For some medications (e.g., antibiotics), it is important that a certain amount of the medication is at work at all times. Such a medication typically has a *demand rate constraint* specified by a 2-tuple (*L, P*): This constraint requires that the total size of dispatches within any interval of length *P* (time units) to be at least equal to the lower bound *L* granules. Such a constraint is also appropriate for produce deliveries. A green grocer would want to have a lower limit, such as "at least 20 crates in any 7 consecutive days" or "at least 20 crates every week". These limits are specified as (20, 7)-*uniform* and (20, 7)-*periodic*, respectively, in the GSS. The definition of demand rate given above is for the (*L, P*)-*uniform* variant. The definition of the periodic variant (*L, P*)-*periodic* segments time into periods of length *P* and requires that the total size of dispatches in any period be at least *L*. This variant is less stringent. In the green grocer example, a schedule that delivers 20 crates on Monday of a week and 20 crates on Friday of the following week satisfies the (20, 7)-*periodic* demand rate constraint but not the (20, 7)-*uniform* constraint. We focus on the uniform variant throughout the paper but will discuss when it is necessary to relax the requirement to the periodic variant. Since there is no source of ambiguity, we will omit mentions of time unit and granularity hereafter.

As stated earlier, a GSS is *consistent* if the constraints defined by the constraint parameters given by the specification are not in conflict with each other. Here, by a *conflict*, we mean some difference or incompatibility in one or more parameters that the scheduler cannot resolve automatically. When given a GSS, the scheduler first checks whether the specification is consistent. It proceeds to check for feasibility and try to compute a feasible schedule if the specification is consistent. Otherwise, it requests conflict resolution, either manually by persons who negotiated the specification or automatically by some specification tools.

The following theorem lists the conditions that constraints must satisfy to be consistent. An inform proof of the theorem can be found in [1].

Theorem 1: *The constraints given by a GSS are consistency if they satisfy all of the following conditions:*

(1) *Valid size range:* $0 \le d_{min} \le d_{max}$

(2) *Valid separation range:* $0 < s_{min} \le s_{max}$

(3) *Valid limit parameters:* $d_{max} \le B$ *and* $s_{max} \le P \le T_{min}$

(4) *Feasible supply rate:* $d_{min} \times Floor[R/s_{max}] \le B$

> ***(5)*** *Feasible demand rate: Ceil[L /d_{max}] $\leq$ Floor[P/s_{min}]*
>
> ***(6)*** *Consistent dispatch rate limits: B/R $\geq$ L/P*

where Ceil[x] and Floor[x] denote the ceiling and floor of x, respectively..

Example 1 below illustrates that the conditions (1) – (6) are not sufficient to ensure feasibility of the GSS in general.

Example 1: [d_{min} , d_{max}] = [5, 6], [s_{min} , s_{max}] = [45, 46], (*B, R*) = (10, 79), (*L*, *P*) = (17, 159)

The parameters satisfy these conditions. However, no schedule can satisfy all the constraints defined by them: It is not possible to meet the demand rate constraint with four dispatches of size 5, because some interval of length 159 contains only three dispatches with a total size 15. If three dispatches are used to satisfy the demand rate constraint, their sizes must be equal to or larger than 5, 6 and 6. But such a schedule cannot meet the supply rate constraint (10, 79) because consecutive dispatches of sizes 5 and 6 or 6 and 6 can be separated by at most 46.

Conditions (1) – (6) are sufficient, however, in the special cases listed in the following corollary.

Corollary 2: *In each of the following special cases, a GSS is consistent and feasible if the constraints it defines satisfy conditions (1)-(6) listed in Theorem 1:*

> *(1)* *L/P = 0, or B/R = ∞*
>
> *(2)* *d_{max} divides B, R divides P, and the separation range includes R d_{max} / B*

Condition (1) means either the demand rate or the supply rate is not constrained. The fact that a consistent specification is also feasible is obvious in this case. Case (2) permits a periodic schedule which uses a constant dispatch size of d_{max} and separations equal to Floor[*R* d_{max} / *B*] and Ceil[*R* d_{max} / *B*] alternately. Such a schedule satisfies both the supply rate and demand rate constraints, as illustrated by Example 2.

Example 2: [d_{min} , d_{max}]= [3, 5], [s_{min} , s_{max}] = [35, 46], (*B, R*) = (10, 79), (*L*, *P*) = (17, 158)

The schedule that makes dispatches of size 5 at 0, 39, 79, 118, 158, and so on satisfies both variants of the demand rate constraint as well as the supply rate constraint.

3 Feasibility Test Based on Minimum Demand Schedule

We describe in this section and the next section the *demand versus supply test* for determining whether a GSS is feasible in the general case. The steps in the test are listed below.

Demand versus Supply Test* (DST)*:

1. *Check whether the given set* $\{[d_{min}, d_{max}], [s_{min}, s_{max}], (T_{min}, T_{max}), (B, R), (L, P)\}$ *of parameters satisfies all the conditions in Theorem 1. If some condition in Theorem 1 is not satisfied, declare the specification inconsistent and return.*
2. *Check whether either condition in Corollary 2 is satisfied. If yes, declare the specification feasible and return.*
3. *Carry out one of the following steps to check the specification for feasibility:*
   (a) *Case* $R \geq P$*: Find the minimum demand schedule of the specification and the required supply of the schedule. If the required supply is no greater than the budget B, declare the specification feasible and return the minimum demand schedule. Else, declare that the specification may be infeasible and return.*
   (b) *Case* $R < P$*: Find the maximum supply schedule of the specification and the available supply of the schedule. If the available supply is no less than L, declare the specification feasible and return the maximum supply schedule. Else, declare that the specification may be infeasible and return.*

This section defines the terms referred to in 3(a) and provides the rationale behind the step. We will define in the next section maximum supply schedule and available supply of the schedule and describe the rationale for Step 3(b).

To define minimum demand schedule, we need a few additional definitions. We call a periodic schedule that has k dispatches per period a *k-dispatch schedule* and denote the total size of dispatches in each period by $\delta(k)$, and the length of the period by $\pi(k)$. Each period of such a schedule can be defined by the sequence $\{(t_0, d_0), (t_1, d_1) \ldots (t_i, d_i,) \ldots. (t_{k-1}, d_{k-1})\}$ of k dispatch times and sizes. t_0 is the start of the period and the time of the first dispatch in the period. Subsequent dispatches in the period are made at times $t_1, t_2 \ldots t_{k-1}$, for $t_0 < t_1 < \ldots < t_{k-1}$, relative to the start of the period. For the sake of convenience, we let t_k denote the end of the period. $t_k = t_0 + \pi(k)$ is the start of the next period; the size d_k of the dispatch at time t_k is d_0. Finally, let $x_i = t_i - t_{i-1}$, for $i = 1, 2 \ldots k$, denote the separations between consecutive dispatches in each period.

For a given consistent set $\{[d_{min}, d_{max}], [s_{min}, s_{max}], (B, R), (L, P)\}$ of constraint parameters, a k-dispatch schedule, defined by $\{(t_0, d_0), (t_1, d_1), \ldots, (t_k, d_k)\}$ and $x_i = t_i - t_{i-1}$, for $0 < i \leq k$, is a *k-dispatch minimum demand schedule* when the dispatch sizes and separations in each period are solutions of the following two-part problem.

k-Dispatch Minimum Demand Problem:

(1) *Find integers* $x_1, x_2 \ldots x_k$ *that maximizes* $\pi(k) = \Sigma_{1 \leq j \leq k}\, x_j$ *subject to the constraints* $s_{min} \leq x_i \leq s_{max}$ *for* $i = 1, 2, \ldots k$ *and* $\pi(k) \leq P$.

(2) *Find integers $d_0, d_1 \ldots d_{k-1}$ that minimizes $\delta(k) = \Sigma_{0 \leq j \leq k-1} d_j$ subject to the constraints $d_{min} \leq d_i \leq d_{max}$ for $i = 1, 2, \ldots k$ and $\delta(k) \geq L$.*

In other words, a *k*-dispatch minimum demand schedule has a longer period and a smaller total dispatch size than any *k*-dispatch schedule that meets the demand rate constraint (L, P). We call the schedule *MinDS*(k) for short.

To illustrate, we consider the following example:

Example 3: $[d_{min}, d_{max}] = [3, 10]$, $[s_{min}, s_{max}] = [45, 55]$, $(B, R) = (29, 220)$, $(L, P) = (17, 158)$

There must be at least 2 dispatches to make up the total size of 17, and there can be at most three dispatches within a demand rate constraint period of 158. A 2-dispatch minimum demand schedule MinDS(2) is given by {(0, 9), (55, 8), (110, 9), …}, that is, $x_1 = x_2 = 55$, $d_0 = 9$, $d_1 = 8$, period $\pi(2) = 110$, and total size $\delta(2) = 17$. A 3-dispatch minimum demand schedule is MinDS(3) = {(0, 6), (52, 6), (105, 5), (158, 6), …} with period $\pi(3) = 158$ and total size $\delta(3) = 17$.

We say that a *k*-dispatch schedule, $\{(0, d_0), (x_1, d_1), (x_1 + x_2, d_2) \ldots (x_1 + x_2 \ldots + x_k, d_0)\}$, with period $\pi(k)$ and total dispatch size per period $\delta(k)$, is *well-formed* if $d_0 \geq d_1 \geq \ldots \geq d_{k-1}$; $x_1 \leq x_2 \ldots \leq x_k$; and for all $0 \leq i \leq k-1$, d_i equals either Floor[$\delta(k)/k$] or Ceil[$\delta(k)/k$], and for all $1 \leq i \leq k$, x_i equals either Floor[$\pi(k)/k$] or Ceil[$\pi(k)/k$]. In other words, according to a well-formed schedule, the sizes of the dispatches differ by at most one, separations between dispatches differ by at most one, and the larger dispatches are make closer together earlier in the period.

The following lemma is true for Example 3. It is true in general because all the coefficients of the terms in the objective functions of the *k*-dispatch minimum demand problem are equal

Lemma 3: *If the k-dispatch minimum demand problem admits a solution, then there is a a well-formed MinDS(k).*

Hereafter, by a MinDS(k), we mean the well-formed *k*-dispatch minimum demand schedule except for where it is stated otherwise.

The *required supply* of a MinDS(k) is the total size of all the dispatches within the replenishment interval [0, R) when $P \leq R < \infty$. A *minimum demand schedule* (*MinDS*) of a general schedule specification is a schedule that has the minimum required supply among MinDS(k) for all feasible values of k.

To illustrate, we return to the specification in Example 3. The replenishment delay R given by the specification is 220. The required supply of the 2-dispatch minimum demand schedule MinDS(2) is $2 \times 17 = 34$. The first period of the 3-dispatch minimum demand schedule MinDS(3) falls within [0, 220). The first two dispatches of size 6 in the second period also fall in this interval. Therefore, the required supply of MinDS(3) is 17 + 12 = 29, and the minimum demand schedule MinDS of this specification is MinDS(3).

Fig. 2 gives a pseudo-code description of the functions for finding separations and dispatch sizes that are solutions of the *k*-dispatch minimum demand problem. The functions take constant time. The worst case time required to find the minimum demand schedule of the specification is in order of the maximum of the widths of the separation range and the dispatch size range.

```
find_MinDS_separations ($s_{min}$, $s_{max}$, P, k, x, period)
{
   // Inputs are separation range, demand rate constraint interval, the
   //  number k of dispatches, pointer to separation array x of size k
   //  and the pointer to period;
   // Output is either succeeded or failed.
   remainders = 0;
   * period = 0;
   if (k * $s_{min}$ > P) goto quit;
   x[0] = P/k;                          //Try first to make the period equal to P.
   if (x[0] > $s_{max}$) {
       *period = k * $s_{max}$ ;           // period is less than P
       for (j = 0; j < k; j = j + 1) {
           x[ j ] = $s_{max}$ ;
       }
    } else {
       *period = k * x[0];
       remainders = P – k * x [0];  // remainders < k
       for (j = 1; j < k; j =  j + 1) {
           x[ j ] = x[0];
       }
       if (x[0] < $s_{max}$) {
           for (j = k - 1; j > k – 1 – remainders; j = J – 1) {
               x[ j ] = x[ j ] +1;
           }
           *period = * period + remainders;
       }
    }
quit:
    if (*period == 0)  return = failed;
    else return = succeeded;
```

```
find_MinDS_dispatch_sizes ($d_{min}$, $d_{max}$, L, k, d, total_size)
{
   // Inputs are dispatch size range, demand size limit, the number k of
   // dispatches, pointer to dispatch size array d of size k and the
   // pointer to total_size;
   // Output is either succeeded or failed.
   remainders = 0;
   * total_size = 0;
   if (k * $d_{max}$ < L) goto quit;
   d[0] = L/k;                          // Try first to make total_size equal to L.
   if (d[0] < $d_{min}$) {
       *total_size = k * $d_{min}$ ;       // total_size is larger than L
       for (j = 0; j < k; j = j+1) {
           d[ j ] = $d_{min}$ ;
       }
    } else {
       remainders = L – k * d [0];   // remainders < k
       if (d[0] == $d_{max}$) && (remainders != 0)) goto quit
       *total_size = k * d[0];
       for (j = 1; j < k; j = J + 1) {
          d[ j ] = d[0];
       }
       if (d[0] < $d_{max}$) {
          for (j = 0; j = remainders – 1; j = j + 1) {
             d [ j ] = d[ j ] + 1;
          }
          *total_size = *total_size + remainders;
       }
    }
quit:
    if (*total_size == 0)  return = failed;
    else return = succeeded;
```

Fig. 2 Algorithm for finding MinDS(*k*)

The following lemma allows us to say that since the budget *B* given by the specification in Example 3 is 29, which equals the required supply of its MinDS, the specification is feasible.

Lemma 4: *A given GSS* {[d_{min}, d_{max}], [s_{min}, s_{max}], (*B*, *R*), (*L*, *P*)}, *where* $R \geq P$, *is feasible if the budget B is equal to or larger than the required supply of the minimum demand schedule MinDS of the specification.*

Proof: By definition, all dispatch(s) within the interval [0, *R*) can be scheduled according to the MinDS if the required supply of the minimum demand schedule is not larger than *B*. We need to show that dispatches after R can also be made according to the schedule. Without loss of generality, let us suppose that the required supply of the MinDS is equal to B. Let (t, d) denote the time and size of the last dispatch before R, and (t', d') be the time and size of the first dispatch at or after R. The initial segment of the schedule is then $\{(0, d_0), (t_1, d_1) \dots (t_{k-1}, d_{k-1}), (\pi(k), d_0), (\pi(k) + t_1, d_1) \dots (t, d), (t', d') \dots\}$.

Also by definition, the current budget immediately after t is zero. At time $R + t_0 < t'$, the current budget is replenished by d_0. Because MinDS is well-formed, d_0 is at least equal to d', making the dispatch (t', d') feasible. The current budget is replenished again $R + t_1, \dots, R + t_{k-1}, R + \pi(k), R + \pi(k) + t_1$ and so on by $d_1, \dots, d_{k-1}, d_0, d_1$, and so on, respectively, just in time and by amounts sufficient for the dispatches after t' to be scheduled according to the MinDS. ■

Fig. 3 illustrates graphically the feasibility the MinDS of the GSS in Example 3. The top time line shows the first 3 and half periods of the schedule: The dark lines at 0, 52, 158, 210, and so on represent dispatches of size 6. Narrow boxes at 105, 263, and so on represent dispatches of size 5. The dotted line illustrates how the current budget varies as dispatches are made. It becomes zero after the dispatch at 210. The chunk of budget of size 6 consumed at time 0 is replenished at time 220. The bottom time line indicates the subsequent replenishments of size 6, indicated by dark lines, and of size 5, indicated by narrow boxes, at times 272 (220 + 52), 325 (220 + 105), …. and so on. We can see that the current budget is always sufficient to meet the demand of the MinDS.

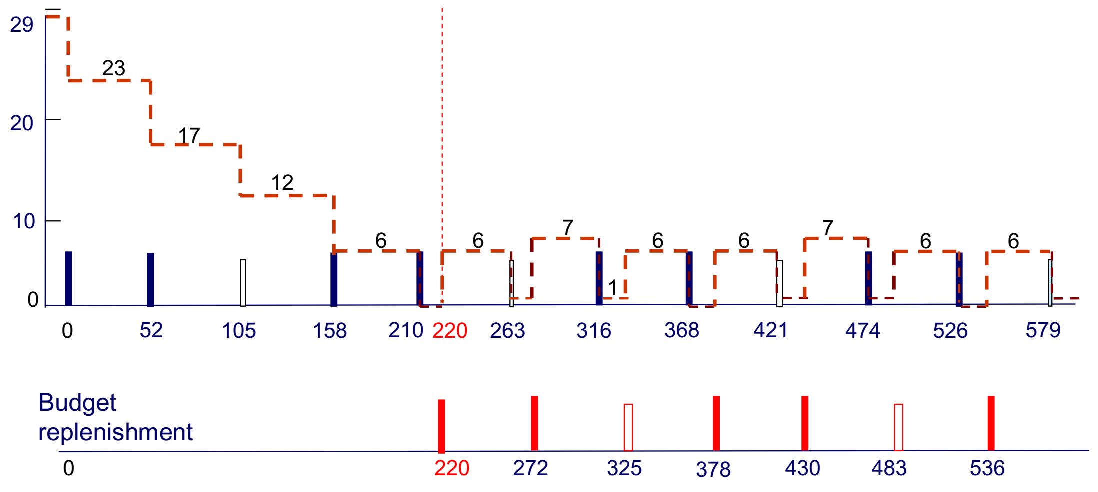

Fig. 3 MinDS of the specification in Example 3

The following corollary states a special case for which the DST test based on Lemma 4 is accurate, i.e., the condition stated by the lemma is both necessary and sufficient.

Corollary 5: *If according to the minimum demand schedule of a GSS* $\{[d_{min}, d_{max}],$

$[s_{min}, s_{max}]$, (B, R), $(L, P)\}$, where $R \geq P$, all dispatches have identical sizes and all pairs of consecutive dispatches have equal separations, then the GSS is feasible if and only if the budget B is equal to or larger than the required supply of the schedule.

A fair question at this point is whether the sufficient condition stated in Lemma 4 is necessary in general. Is it possible for a GSS to have a feasible schedule when the budget B is less than the required supply of the specification? To get some insight to this question, we consider the GSS in the following example:

Example 4: $[d_{min}, d_{max}] = [2, 3]$, $[s_{min}, s_{max}] = [5, 5]$, $(L, P) = (10, 20)$, $(B, R) = (B, 26)$

The replenishment delay is 26. We want to determine the minimum value of B for this specification to be feasible. The top time line in Part (a) of Fig. 4 shows a well-formed minimum demand schedule: It is a 4-dispatch schedule with period 20 and dispatches of sizes 3, 3, 2 and 2. (Again, dark lines and narrow boxes represent dispatches of sizes 3 and 2, respectively.) The required supply is 16. Lemma 4 tells us that the specification is surely feasible if B is equal to or larger than 16. The schedule in Part (a) supposes that B is equal to 16. After the dispatch of size 3 is made at time 25, the current budget is exhausted. Three units of budget are replenished at time 26. This replenishment, followed by another three units at time 31 and subsequent replenishments at times indicated by the bottom time line, allows the periodic schedule to continue as shown.

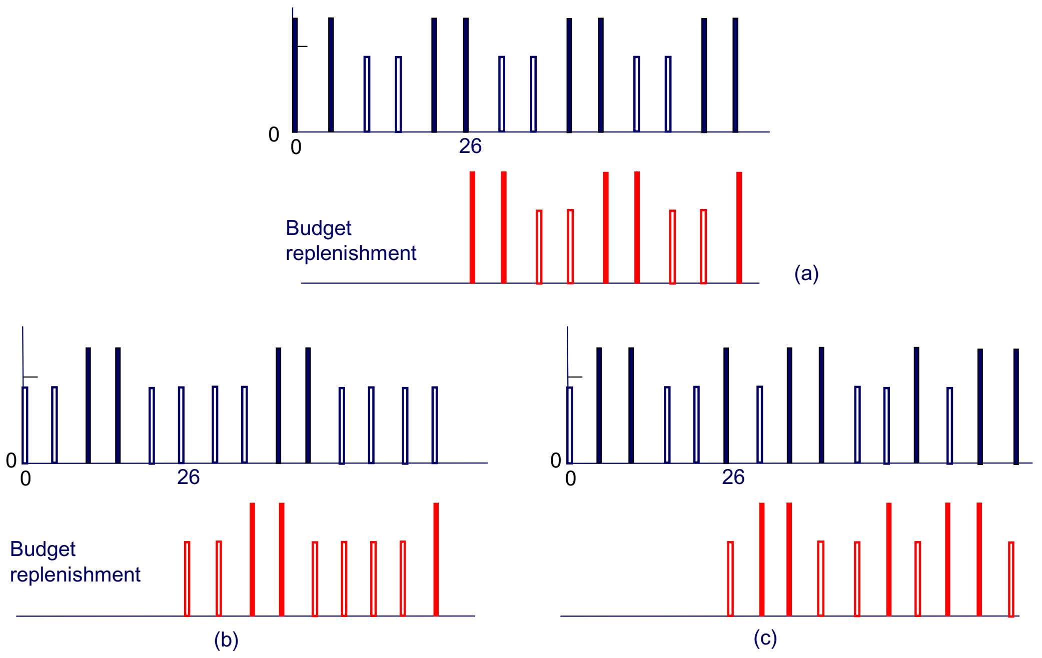

Fig. 4 Example illustrating the tightness of the condition in Lemma 4

The top time line in Part (b) of Fig. 4 shows the initial segment of another 4-dispatch minimum demand schedule. Unlike the well-formed MinDS in Part(a), dispatches of size 2 are made before the dispatches of size 3 according to this schedule: The initial segment is given by the 2-tuples (0, 2), (5, 2), (10, 3), (15, 3), (20, 2) and (25, 2). A budget of 14 is sufficient for these dispatches. Suppose that B is equal to 14. Just prior to the first replenishment at 26, the current budget is 0. The amounts of budget replenished at 26 and 31 are 2, however. They are insufficient to support two dispatches of size 3 at 30 and 35, forcing the sizes of dispatches at these time instants to be 2. The dispatch schedule allowed by the supply rate of (14, 26) is shown in part (b). Unfortunately, it satisfies neither variants of the (10, 20) demand rate constraint. Part (c) shows yet another 4-dispatch minimum demand schedule: Dispatches of sizes 2, 3, 3, 2, 2, and 3, are made at time 0, 5, 10, 15, 20 and 25. So, suppose that B is equal to 15, sufficient for these dispatches. The bottom time line shows the replenishments starting at time 26. The resultant schedule is given by the top time line. This schedule also does not satisfy the uniform variant of the demand rate constraint (10, 20), but it satisfies the periodic variant.

The condition stated in Lemma 4 appears to be tight for Example 4. Our conjecture is that it is both sufficient and necessary in general, even when dispatches have different sizes and separations according to minimum demand schedule(s) of the specification. Lemma 6 stated below is a weaker statement on the accuracy of the test.

***Lemma 6**: A GSS* $\{[d_{min}, d_{max}], [s_{min}, s_{max}], (B, R), (L, P)\}$*, where* $R \geq P$*, has no periodic feasible schedule if the required supply of its MinDS is larger than B.*

Proof: Suppose that a GSS has a periodic feasible schedule F, but the test according to Lemma 4 fails. In other words, the required supply Σ of the MinDS is larger than the budget B.

In contrast, the fact that F is feasible implies that the required supply Σ_F (i.e., the total size of all dispatches in [0, R)) of F is at most equal to B. Let n denote the number of dispatches per period according to F, δ_F denotes the total size of the n dispatches in each period, and π_F denotes the length of the periods. Let m (= Floor [R/π_F]) be the number of periods that lie entirely within the replenishment interval [0, R), and $t = m\ \pi_F$ is the end of the latest of these periods. The top time line in Fig. 5 illustrates the relationship between these time instants.

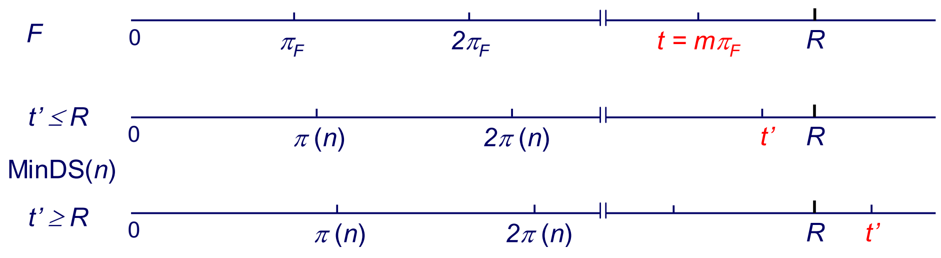

Fig. 5 Periods of *F* and MinDS(*n*)

The fact that F has n dispatches per period implies that the n-dispatch minimum demand problem has a solution MinDS(n). We only need to consider the case that MinDS(n) is different from F, since MinDS(n) being identical to F contradicts our supposition.

By definition of MinDS(n), the required supply $\Sigma(n)$ of MinDS(n) is at least equal to the required supply of MinDS. It follows from our supposition that $\Sigma(n) \geq \Sigma > B$. Furthermore, the total dispatch size $\delta(n)$ and period $\pi(n)$ according to MinDS(n) is such that $\delta(n) > L$, $\pi(n) < P$, $\pi_F \leq \pi(n)$, and $\delta_F \geq \delta(n)$. We can, therefore, conclude that the end $t' = m\ \pi(n)$ of the m-th period according to MinDS(n) is equal to or later than t. The bottom two time lines in Fig. 5 illustrate the two cases, $t' < R$ and $t' \geq R$, when t' is compared with R:

In the former case, the fact $R - t \geq R - t'$ follows from the inequality $\pi_F \leq \pi(n)$. Let ε_F denote the total size of all dispatches in the interval $[t, R)$ according to F. $\varepsilon(n)$ denotes the total size of all dispatches in $[t', R)$ according to MinDS(n). By supposition $\Sigma(n) > \Sigma > \Sigma_F$ and the property $\delta_F \geq \delta(n)$ of MinDS(n), we have $\varepsilon(n) > \varepsilon_F$. Since the schedules are periodic, this inequality implies that the total size of dispatches in the initial segment $[0, R - t)$ according to F is smaller than the total size of dispatches in the initial segment $[0, R - t')$ according to MinDS(n). This fact in turn implies that the total budget (being at most equal to ε_F) replenished according to F in $[R, 2R - t)$ is less than the total budget replenished according to MinDS(n) in $[R, 2R - t')$. The smaller replenishment in turn leads to a smaller total size of dispatches scheduled in $[R, 2R - t)$ according to F than the total size of dispatches scheduled in $[R, 2R - t')$ according to MinDS(n). Again, because the schedules are periodic, we can conclude that the total size of dispatches scheduled in $[R - t, 2R - 2t)$ (and the budget replenished in $[2R - t, 3R - 2t)$) according to F is smaller than the total size of dispatches scheduled in $[R - t', 2R - 2t')$ (and the budget replenished in the interval $[2R - t', 3R - 2t')$ of length $R - t'$) according to MinDS(n). Repeating this argument, we eventually come to the conclusion that the total dispatch size δ_F per period according to F is less than the total dispatch size $\delta(n)$ according to MinDS(n). The conclusion contradicts the definition of n-dispatch minimum demand schedule.

In the case of $t' \geq R$, we must have $\Sigma(n) \leq \Sigma_F \leq B$. This contradicts our supposition that $\Sigma(n) > \Sigma > B$. Since both cases lead to contradictions, the supposition that a GSS has a periodic feasible schedule when it fails the DST test must be false. ■

4 Feasibility Test Based on Maximum Supply Schedule

When the replenishment delay is smaller than the demand rate constraint interval, the demand versus supply feasibility test first constructs a (periodic) maximum supply schedule. A k-dispatch schedule, defined by $\{(t_0, d_0), (t_1, d_1) \ldots (t_k, d_k)\}$ and $x_i = t_i - t_{i-1}$ for $0 < i \leq k$, is a *k-dispatch maximum supply schedule* for a given set $\{[d_{min}, d_{max}], [s_{min}, s_{max}], (B, R), (L, P)\}$ of constraint parameters when the dispatch sizes and separations in each period are solutions of the following two-part problem.

k-Dispatch Maximum Supply Problem:

(1) *Find integers $x_1, x_2 \ldots x_k$ that minimize $\pi(k) = \Sigma_{1 \leq j \leq k} x_j$ subject to the constraints $s_{min} \leq x_i \leq s_{max}$ for $i = 1, 2, \ldots k$ and $\pi(k) \geq R$.*

(2) *Find integers $d_0, d_1 \ldots d_{k-1}$ that maximizes $\delta(k) = \Sigma_{0 \leq j \leq k-1} d_j$ subject to the constraints $d_{min} \leq d_i \leq d_{max}$ for $i = 1, 2, \ldots k$ and $\delta(k) \leq B$.*

We call the schedule *MaxSS*(k) for short. The schedule has a shorter period and a larger total dispatch size than any k-dispatch schedule that meets the supply rate constraint (B, R). Because the coefficients of objective functions are equal to 1, we can confine our attention to well-formed schedules as we did in the case of minimum demand schedules.

Corollary 7: *If the k-dispatch minimum supply problem admits a solution, then there is a well-formed k-dispatch maximum supply schedule MaxDS(k).*

Like the k-dispatch minimum demand schedule, the well-form k-dispatch maximum supply schedule can be found in constant time in a way similar to that described in Fig. 2.

To illustrate, we return to Example 1. We note that the constraint $\pi(k) \geq 79$ *and* $\delta(k) \leq 10$, together with size and separation ranges being [5, 6] and [45, 46], respectively, means that $k = 2$ is the only choice. The MaxSS(2) is given by {(0, 5), (45, 5), (90, 5), (135, 5), (180, 5) …}. Its period is $\pi(2) = 90$ and total dispatch size is $\delta(2) = 10$. This schedule does not meet the demand rate constraint (L, P) = (17, 159)-*uniform*, because the total size of all dispatches within the time interval [1, 160) is only 15, less than the required minimum total size of 17. To determine whether (17, 159)-periodic constraint is satisfied, we segment time into periods of length 159. The schedule does not meet the periodic variant of (17, 159) constraint because one of the periods falls in the interval [636, 795) and the total size of dispatches in that interval is only 15. Since even the maximum supply schedule fails in this respect, we can conclude that the specification in this example is not feasible as we did in the previous section.

The test used on Example 1 can be stated in general in terms of the maximum supply schedule of the specification and the available supply of the schedule: The *available supply* of a k-dispatch maximum supply schedule is the minimum of the total sizes of dispatches scheduled according to the MaxSS(k) in demand rate constraint intervals [t, $t + L$) for all $t \geq 0$ in the duration of the schedule. The *maximum supply schedule* (MaxSS) of the given specification is a schedule that has the maximum available supply among MaxSS(k) for all feasible values of k.

The available supply of MaxSS(2) in the previous example is the total size of dispatches in the interval [1, 160), which is equal to 15. In this case, MaxSS(2) is the MaxSS. Similar to MinDS presented in the previous section, the worst case time required to find the MaxSS of a specification is in order of the maximum of widths of separation and dispatch size ranges.

Below is another illustrative example. It is similar to Example 1, except that the ranges of both dispatch size and separation are wider.

Example 5: $[d_{min}, d_{max}] = [3, 10]$, $[s_{min}, s_{max}] = [45, 80]$, $(B, R) = (10, 79)$, $(L, P) = (17, 159)$

For this specification, the k-dispatch maximum supply problem has a solution for $k = 1$. MaxSS(1) is defined by {(0, 10), (79, 10), (158, 10), ….}; $\pi(1) = 79$ and $\delta(k) = 10$. The problem also admits a solution for $k = 2$: MaxSS(2) is {(0, 5), (45, 5), (90, 5), (135, 5), …}. The available supply of MaxSS(1) is 20 while the available supply of MaxSS(2) is 15. Hence the MaxSS is MaxSS(1). The following lemma allows us to say that this specification is feasible because its available supply is no less than the lower bound $L = 17$.

Lemma 8: *A given GSS* $\{[d_{min}, d_{max}], [s_{min}, s_{max}], (B, R), (L, P)\}$, *where* $R < P$, *is feasible if the lower bound L of total size in any demand rate constraint interval is at most equal to the available supply of the maximum supply schedule MaxSS of the specification.*

The proof, being similar to that of Lemma 4, is omitted here: The fact that the condition stated by the lemma is sufficient to ensure feasibility follows directly from the definitions of maximum supply schedule and its available supply.

Like the question regarding the condition stated in Lemma 4, we do not know for sure whether the condition stated in Lemma 8 is necessary. Our conjecture is that a general GSS does not have a feasible schedule if its available supply is smaller than the lower bound L given by the demand rate constraint. However, as in the case of $R > P$, the condition is both necessary and sufficient for the existence of periodic feasible schedules. The proof of this statement is similar to that of Lemma 7. The following theorem on DST follows from the above lemmas and corollaries.

Theorem 9: *A given GSS* $\{[d_{min}, d_{max}], [s_{min}, s_{max}], (B, R), (L, P)\}$ *is feasible if it passes the demand versus supply test. It has no feasible periodic schedule if it fails the test.*

5 Heuristic Scheduling Algorithms

We recall that if a GSS passes the DST test, the feasible dispatch schedule found by the scheme is either a minimum demand schedule or a maximum supply schedule. The total size of dispatches in each demand rate constraint interval or supply rate constraint interval is either minimized or maximized. In this respect, the schedule may not be ideal for some applications. Indeed, as performance data presented in the next section will show, a schedule thus found is rigid, as it offers little tolerance to deviation from scheduled dispatch times. This is the reason that we designed the families of heuristic dispatch scheduling algorithms listed in Table 1. (They are called dosage selection algorithms when used for choosing dose sizes and scheduling the doses of a medication [1].) All the algorithms take constant time and generate periodic schedules.

TABLE 1 Dispatch scheduling algorithms

Independent		Rate-Guided	
Maximum	Average	MaxS_AMAP	MaxS_ALAP
Minimum	Uniform	MinS_AMAP	MinS_ALAP
Likely_Large	Random_Large	MaxD_AMAP	MaxD_ALAP
Likely_Small	Random_Small	MinD_AMAP	MinD_ALAP
		AveS_AMAP	AveS_ALAP
		AveD_AMAP	AveD_ALAP

Independent algorithms listed in the left half of the table make independent choices of the sizes and separations for individual dispatches. The four in the first column choose boundary values: Their choices (d, s) of dispatch size and separation are (d_{max}, s_{min}), (d_{min}, s_{max}), (d_{max}, s_{max}) and (d_{min}, s_{min}), respectively, for all dispatches. The Average Algorithm uses average dispatch size and average separation. The last three in this group make independent random choices for each dispatch. Uniform Algorithm chooses for d and s from uniform distributions over the respective ranges of the parameters. Random_Large and Random_Small randomly select d and s from right triangle shape probability density functions over the dispatch size and separation ranges. For the former, the right angles of the dispatch size and separation probability density functions are at d_{max} and s_{min}, respectively, while for the latter, they are at d_{min} and s_{max}.

In contrast, *rate-guided algorithms* make correlated choices of dispatch size and separation. Each of these algorithms starts by fixing one of these constraint parameters and then uses either supply rate constraint or demand rate constraint to guide the choice of the second parameter. The first parts in the names of these algorithms tell us their choices of the first parameter: *MaxS*, *MinS*, *MaxD*, *MinD*, *AveD* and *AveS* indicate that the choices of the algorithms are $s = s_{max}$, $s = s_{min}$, $d = d_{max}$, $d = d_{min}$, $d = (d_{max} + d_{min})/2$, and $s = (s_{max} + s_{min})/2$, respectively.

The algorithms with *AMAP* (*As Much As Possible*) in their names use the supply rate (B, R) to guide the choice of the second constraint parameter. In essence, the algorithms try to make the total dispatch size in each interval of length R as close to the upper limit B as possible, hence the name. These algorithms resemble maximum supply scheduling. When the dispatch size d is chosen first, the algorithms chooses as separation s the value closest to Ceil[R / Floor[B/d]] in the separation range. If s is first chosen, d is the dispatch size closest to Floor[B / Ceil[R/s]]. Algorithms with *ALAP* (*As Little As Possible*) in their names use the demand rate (L, P) as the guide. They try to make the total size of dispatches in each interval P as close to the lower limit L as possible. In this sense, ALAP algorithms do minimum demand scheduling. If s is first chosen, then d is the dispatch size closest to Ceil[L / *Floor*[P/s]]. If d is first chosen, s is the separation closest to Floor[P / Ceil[L/d]].

6 Relative Performance

To compare the relative performance of the heuristics in Table 1 with each other and with the DST scheme, we implemented the algorithms and used them in a simulation experiment to find feasible schedules for synthetic general schedule specifications: The parameters that define the dispatch size, separation and rate constraints in each sample specification were generated randomly. Specifically, separation and interval parameters s_{min}, s_{max}, R *and* P were chosen independently from the uniform distribution [0, 20160]. (If time granularity is 30 seconds, the maximum value 20160 of a separation or interval parameter is equal to seven days. If time granularity is 10 minutes, then the maximum value is 140 days.) Parameters d_{min}, d_{max}, B, and L constraining sizes of individual dispatches and the total size of dispatches in a rate constraint interval were chosen from the uniform distributions [0, 50]. After each sample GSS was generated, we tested it for consistency, i.e., whether its constraint parameters satisfy all the necessary conditions stated in Theorem 1. The sample is discarded if it fails the test. 53.69% of thus generated GSS samples passed the test. The data were collected from a sufficient number of samples to make the statistical error in each of the estimated performance measures less than 2%.

The performance data thus obtained are summarized in Fig.6. We compare the heuristics with each other and with the DST scheme according to three criteria. First, *success rate* of an algorithm is the fraction of all consistent specifications for which the algorithm succeeded in finding a feasible schedule. This criterion is commonly used to measure the effectiveness of suboptimal scheduling algorithms.

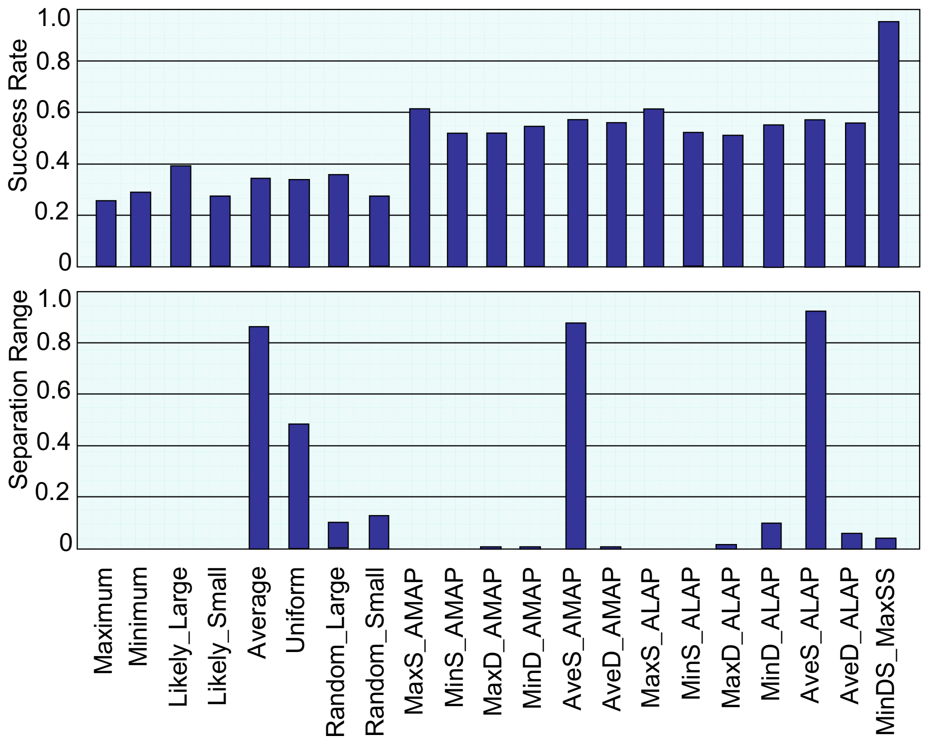

Fig 6 Relative performance of dispatch scheduling algorithms

The other two criteria, maximum allowed tardiness and usage separation range, measure the quality of the resultant schedule. To define them, let ϕ denote the time selected by an algorithm for a dispatch. Let ϕ_{latest} ($>\phi$) denote the latest time to which the dispatch can be postponed without violating separation and rate constraints. The *maximum allowed tardiness* (*MAT*) of the dispatch is the difference $\phi_{latest} - \phi$ between these time instants. As an example, suppose that the dispatches are dispensing of doses of a medication. Then the user can be late taking the current dose by as much as the MAT of the dose. The scheduler must re-compute the schedule and reschedule the current and subsequent doses in order to avoid non-compliance whenever the user is tardy by more than this amount. The *MAT of a schedule* is the minimum of the MAT of all dispatches according to the schedule. Clearly, the larger the MAT of a schedule, the more flexible is the schedule, and the better is the algorithm that finds the schedule.

Let $\phi_{earliest}$ ($<\phi$) denote the earliest time at which the current dispatch can be made without violating any constraint. The difference $\phi_{latest} - \phi_{earliest}$ is called the *usable separation range* with respect to the dispatch. The *usable separation range of a schedule* is the minimum usable separation ranges with respect to all dispatches according to the schedule. This deviation from nominal dispatch time quantifies the leeway the schedule allows in the actual time for the dispatch to be made. The larger the allowed deviation, the better is the schedule.

The performance summary in Fig.6 lists the success rates and usable separation ranges of the heuristics in Table 1 and the DST algorithm (labeled as MinDS-MaxSS). Separation range is normalized with respect to the width of the nominal separation range given by the GSS. The reason for omitting data on MAT is that the usable separation range of each dispatch is twice the MAT of the dispatch, and hence, the usable separation range of a schedule is twice the MAT allowed by the schedule.

Fig. 6 shows that according to success rate, rate-guided algorithms perform significantly better than independent algorithms. The DST scheme has a success rate of 94%. None of the tested heuristic algorithms succeeded in finding feasible schedules of consistent specifications which failed the DST test. Since all the tested heuristic algorithms generate only periodic schedules, this result on the DST scheme follows directly from its optimality among this class of algorithms, as stated in Theorem 9. Nevertheless, the fact that its success rate is over 30% better than the best of these algorithms is somewhat surprising.

It is quite a different story, however, when the algorithms are compared according to usable separation range. As expected, algorithms that use minimum or maximum separation as separation parameter offer zero usable separation range. These algorithms are Maximum, Minimum, Likely-large, Likely-small, MaxS_AMAP, MaxS_ALAP, MinS_AMAP, and MinS_ALAP. The DST schemes does not fare much better in this respect either. This means that the minimum demand or maximum supply schedules of most specifications offer the user with little or no flexibility. For maximum flexibility, we should use algorithms that use average separation whenever the algorithms can find a feasible schedule. A conclusion is, therefore, AveS_AMAP and AveS_ALAP offer the best compromise.

7 Summary

We described in previous sections periodic algorithms for scheduling dispatches according to general schedule specifications that constrain the sizes and separations of the dispatches. This class of algorithms is said to be periodic because the schedules produced by them are periodic. The constraints considered here are ranges of dispatch size and separation of individual dispatches, as well as upper and lower limits of total size of dispatches in specified time intervals. The running time of the DST (demand versus supply test) scheme is in order of the larger of the width of the size range and the width of the separation range. The others take constant time.

The DST scheme is optimal among periodic algorithms from the view point of effectiveness: If the scheme fails to find a feasible schedule for a GSS, then no periodic algorithm can find a feasible schedule. The success rate of the scheme for randomly generated sample specifications is 94%, much better than other heuristic periodic algorithms that were evaluated. Unfortunately, the schedules produced by the DST scheme are poor from the user's point of view. They offer little or no tolerance to timing deviation: Some constraint(s) may be violated when some dispatch is made later or earlier than the time chosen by the schedule. We quantify this aspect of quality by usable separation range or maximum allowed tardiness. When compared according to these criteria, AveS_AMAP and AveS_ALAP algorithms are the best. For each dispatch, these algorithms first choose as the separation from the previous dispatch the average of s_{min} and s_{max}. AveS_AMAP chooses the maximum dispatch size that meets the supply rate (*B, R*) constraint, while the AveS_ALAP choose the minimum dispatch size that meets the demand rate (*L, P*) constraint. These algorithms allow each dispatch to be late (or early) by over 40% of the separation range. On the other hand, they may miss existing feasible periodic schedules by almost a third of the time.

Whether the DST scheme is optimal amongst all scheduling algorithms, including algorithms that produce non-periodic schedules, remains to be a question. Settling this question with a proof of the affirmative or a counter example is part of our current work.

We have confined our attention to the case where the size and separation ranges are contiguous. For some applications, the separation between a pair of dispatches may be in one of many disjoint ranges. The problem of finding feasible schedules becomes considerable more complex with this generalization. Along another dimension, there may be multiple streams of dispatches to be scheduled, each of which is constrained by a different set of parameters. As examples, the scheduler of a medication dispenser often needs to schedule doses of multiple medications, and a supplier may need to schedule the deliveries of multiple produces. It is desirable to bunch dispatches from different streams. This means that the users take medications a fewer number of times per day, and in the case of produce delivery, the supplier runs to the grocer less frequently. We are developing algorithms that yield good quality in this respect.

Acknowledgements

The authors wish to thank Professor C. S. Shih and Mr. H. C. Yen for their valuable suggestions and Mr. Y. F. Chen for his help with simulation. The work is partially supported by Taiwan Academia Sinica thematic project SISARL.

References

[1] P. H. Tsai, H. C. Yeh, C. Y. Yu, P. C. Hsiu, C. S. Shih and J. W. S. Liu, "Compliance Enforcement of Temporal and Dosage Constraints," Proceedings of I*EEE Real-Time Systems*

Symposium, December 2006

[2] C. Hsiu, H. C. Yeh, P. H. Tsai, C. S. Shih, D. H. Burkhardt, T. W. Kuo, J. W. S. Liu, T. Y. Huang, “A General Model for Medication Scheduling,” Institute of Information Science, Academia Sinica, Taiwan, Technical Report TR-IIS-05-008, July 2005

[3]. P. H. Tsai, H. C. Yeh, P. C. Hsiu, C. S. Shih, T. W. Kuo, J. W. S. Liu, “A Scarce Resource Model for Medication Scheduling,” Institute of Information Science, Academia Sinica, Taiwan, Technical Report TR-IIS-06-003, April 2006.

[4] J. W. S. Liu, *Real-Time Systems*, Chapters 6 and 7, Prentice Hall, 2000.

[5] J. P. Lehoczky, L. Sha, and Y. Ding, “The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Behavior,” *Proceedings of IEEE Real-Time Systems Symposium*, December 1989.

[6] N. Audsley, A. Burns, K. Tindell, M. Richardson, and A. Wellings, “Applying a New Scheduling Theory to Static Priority Preemptive Scheduling,” *Software Engineering Journal,* Vol. 5, No. 5, 1993.

[7] C.C. Han, K.J. Lin and J.W. S. Liu, "Scheduling Jobs with Temporal Distance Constraints," *SIAM Journals on Computing,* 1995.

[8] M. Y. Chan and F. Chin, “Schedulers for larger classes of pin-wheel instance,” *Algorithmica,* 9:425--462, 1993.

[9] Z. L. Chen and G. Pundoor, “Order Assignment and Scheduling in a Supply Chain,” *Operations Research,* Vol. 54, No. 3, May-June 2006.

[10] B. Spruri, L. Sha, J. P. Lehoczky, “Aperiodic task Scheduling for Hard Real-Time Systems,” *Real-Time Systems Journal*, Vol. 1, No. 1, 1989.

[11] T. M. Ghazalie and T. P. Baker, “Aperiodic Servers in Deadline Scheduling Environment,” *Real-Time Systems Journal*, 1995.

[12] T. S. Tia, J. W. S. Liu and M. Shankar, “Algorithms and Optimality of Scheduling Aperiodic Requests in Fixed Priority Preemptive Systems,” *Real-Time Systems Journal*, Vol. 10, No. 1, January 1996.