中央研究院
資訊科學研究所
Institute of Information Science, Academia Sinica • Taipei, Taiwan, ROC

TR-IIS-08-001

Algorithms for Scheduling Interaction Medications

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

June 5, 2008 || Technical Report No. TR-IIS-08-001

http://www.iis.sinica.edu.tw/page/library/LIB/TechReport/tr2008/tr08.html

Institute of Information Science, Academia Sinica

Technical Report TR-IIS-08-001

Algorithms for Scheduling Interaction Medications

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

Abstract

This report presents two families of algorithms for scheduling medications that interact with each other. All algorithms accept as input medication directions that have been compiled into a machine readable form. One family of algorithms is called *One-Medication-at-a-Time*, or *OMAT*. As the name implies, an OMAT algorithm produces a full schedule for each of the user's medication in turn. Algorithms in the other family are called *One-Dose-at-a-Time* (*ODAT*) algorithms. An ODAT algorithm schedules one dose at a time without prior knowledge, or with limited knowledge, about future doses. It may fail to make good scheduling decisions compared with OMAT algorithms but can better accommodate dynamic variations in user behavior. Both OMAT and ODAT algorithms apply a variety of priority assignments as well refinements such as letting the user take medications as soon as possible or as late as possible. Data presented here on performance of the scheduling algorithms in terms of success rate and schedule quality can help builders of smart medication dispensers and scheduling tools choose among algorithms and tradeoff merits along different dimensions.

---

* 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

Statistics on health care quality indicates that medication misuse has led to loss of thousands of lives and billions of dollars each year in United States alone [1-6]. By medications here, we mean prescription medications, over-the-counter drugs and nutritional supplements. Medication administration is the last stage of medication process. Administration errors due to failures to follow medication directions contribute approximately 25 to 40 percent of all medication errors and major causes of administration errors include user's misinterpretation of directions and forgetfulness and inability to follow rigid schedules. This fact has motivated the advent and use of medication dispensers and scheduling tools that are designed to help users in compliance to directions (e.g. [7, 8]). An automatic (medication) dispenser such as the one described in [7] relieves the user of the burden to manually interpret medication directions, schedules the user's medications, reminds the user at times when medications should be taken, controls the dosages dispensed each time, dynamically readjusts the medication schedule to stay compliant when the user is tardy, and when compliance becomes impossible, sends appropriate notifications to designated person(s) or tool(s). Scheduling tools [8] are software programs that can be installed in devices such as medication dispensers or PDA or are provided as web services to help the user with medication scheduling.

The direction of a medication specifies constraints on how the medication should be administered. Constraint parameters include ranges of dose size, temporal separation between doses, maximum and minimum dosage over some time intervals, and so on. Some medications interact with each other or with food. Directions of interacting medications impose additional constraints on how the medications are to be scheduled when the user is taking them at the same time. Scheduling flexibility comes from the sizable ranges in constraint parameters typically provided by directions of modern medications. By exploiting this flexibility, an automatic medication dispenser or scheduling tool can give its user more leeway in response to reminders and thus make the medication schedules it produces user friendlier. This paper focuses on scheduling algorithms that a dispenser or tool needs for such exploitation.

Specifically, we present here two families of algorithms for scheduling interacting medications. All algorithms accept as input medication directions that have been compiled into a machine readable form, called *medication schedule specification* (*MSS*) [9, 10]. In particular, the MSS for a user specifies requirements and constraints derived from directions of all medications taken by the

user. One family of algorithms is called *One-Medication-at-a-Time*, or *OMAT*. As the name implies, an OMAT algorithm produces a full schedule for each of the user's medication in turn. Because they are off-line, OMAT algorithms can be more complex. They tend to produce better schedules. Algorithms in the other family are called *One-Dose-at-a-Time* (*ODAT*) algorithms. An ODAT algorithm schedules one dose at a time with no prior knowledge or limited knowledge about future doses. It may fail to make good scheduling decisions compared with OMAT algorithms but can better accommodate dynamic variations in user behavior. Both OMAT and ODAT algorithms apply a variety of priority assignments including the Earliest-Deadline-First (EDF) and Rate-Monotonic (RM) schemes [11]. The algorithms also incorporate refinement strategies such as letting the user take medications as soon as possible or as late as possible.

We measure the performance of a medication scheduling algorithm in terms of success rate and schedule quality. The *success rate* of an algorithm is the percentage time the algorithm succeeds in finding a schedule of user's medications meeting all hard constraints specified by the given MSS. Schedule quality is measured in terms of several criteria, including percentage deviation and normalized allowed tardiness. Roughly, the former measures the rigor in compliance while the latter measures user friendliness of the schedule. We will return later to define these metrics precisely, as well as the division of constraints into hard and firm types. The performance data on different priority assignments and refinements presented here can help builders of smart dispensers and scheduling tools choose among the scheduling algorithms and tradeoff their merits along different dimensions.

In the remainder of the paper, Section 2 presents the scheduling constraints given by MSS. Section 3 describes the major steps in medication scheduling and the underlying resource model. It also describes an algorithm for selecting dose sizes of individual medications and defines the performance metrics mentioned earlier. Section 4 describes basic and enhanced versions of algorithms in the OMAT and ODAT families. Section 5 describes simulation experiments conducted for the purpose of their evaluation and the performance data obtained from the experiments. Section 6 summaries our results and presents future works.

2 Medication Schedule Specification

Again, the constraints of medication schedules defined by the user's MSS capture the directions of user's medications. Medication directions can be in various forms: hand written or printed, in natural language or tabular form, on-line from some prescription order entry system

(POES) or drug libraries and so on. Without loss of generality, we assume that the user's pharmacist has used a transcription tool such as the one described in [12] and translated and compiled the directions into a machine-readable specification after making sure that all effects of drug interactions have been properly taken into account. Hereafter, we speak solely of medication dispensers and their schedules. Clearly, all our discussions on algorithms and their performance apply equally well to scheduling software tools.

The specific about the data structure of MSS provided to the dispenser is unimportant for our discussion here. It suffices to say that MSS is composed of two parts: dosage parameters and special instructions. For each medication taken by the user, the *dosage parameters* (*DP*) part contains constraints on how the medication is to be administrated when the medication is taken alone. If the medication interacts with any other medication taken by the user or with food, the *special instructions* (*SI*) part specifies how the administration of the medication is to be modified because of the interaction.

2.1 Dosage Parameters

Figure 1 lists the constraint parameters in the DP part for a medication named *M*. The medication comes in granules of size *g*. *g* is the smallest dose size in milligram, tablet, or some other unit. In subsequent discussions, dose sizes have integer values: By dose size *k*, we mean *k* granules of the specified granularity *g*. Similarly, time instants have integer values: Time instant *j* means *j* times the time granularity used by the scheduler from the start of the schedule. Hereafter, we use one-hour as time granularity except where it is stated otherwise.

M:	Name of the medication
g:	Granularity
[T_{min}, T_{max}]:	Minimum and maximum duration over which the medication is administered
[d_{min}, d_{max}]:	Nominal minimum and maximum dose size
[s_{min}, s_{max}]:	Nominal minimum and maximum separations between (consecutive) doses
[D_{min}, D_{max}]:	Absolute minimum and maximum dose sizes
[S_{min}, S_{max}]:	Absolute minimum and maximum separations
(*B*,*R*):	Maximum intake over a specified time interval given by budget *B* and replenishment delay *R*
(*L*,*P*):	Minimum intake over a specified time interval given by lower bound *L* and interval length *P*

Figure 1 Dosage parameters

Nominal dose size range [d_{min}, d_{max}] and *nominal separation range* [s_{min}, s_{max}] listed in lines 4

and 5 in Figure 1 say that, usually, the size of each dose should be in the range delimited by *nominal minimal dose size* d_{min} and *maximum dose size* d_{max} and time separation between consecutive doses should be in the range delimited by the *nominal minimum separation* s_{min} and *maximum separation* s_{max}. Take Vicodin [13] as an example. The medication is a combination of painkiller and cough suppresser. Its direction says "*All forms of Vicodin are taken 4 to 6 hours as needed for pain. The usual dose of Vicodin is 1 or 2 tablets, up to a maximum of 8 tablets per day.*" Hence, the DP part of Vicodin specifies the nominal dose size range [1, 2] and nominal separation range [4, 6].

Constraints such as "*up to a maximum of 8 tablets per day*" in direction of Vicodin are specified in terms of supply rate: *Supply rate* (*B, R*) constraints the total size of all doses over any time interval of length *R* to be no more than *B*. So, the supply rate of Vicodin is (*B, R*) = (8, 24). Similarly, the *demand rate (L, P*) listed in line 9 in Figure 1 requires that the total size of all doses in every interval of length *P* to be at least equal to *L.* This constraint ensures that a certain amount of the medication is at work at all times. As examples, medications used for treatments of infections and high blood pressure typically have demand rate constraints.

It is not always possible, and often not necessary, for the scheduler to keep dose sizes and separations within their nominal ranges. *Absolute dose size range* [D_{min}, D_{max}] and *absolute separation range* [S_{min}, S_{max}] intend to provide scheduling flexibility when it is safe to do so. In general, [d_{min}, d_{max}] and [s_{min}, s_{max}] are contained in [D_{min}, D_{max}] and [S_{min}, S_{max}], respectively. For example, the direction of Fosamax [13] says: *"The usual dose of Fosamax is 10 milligram once a day or 70 milligram once a week*". Its nominal dose range and separation range are [1, 1] and [24, 24], respectively, for granularity of 10-milligram tablet. Its absolute dose size and absolute separation ranges are [1, 7] and [24, 168], respectively. It supply rate is (*B*, *R*) = (7, 168).

We note that some constraints specified by the DP part of MSS are firm, and some are hard. Specifically, nominal dosage parameters (i.e. dose size and separation ranges) are *firm constraints*: The scheduler tries to meet them whenever possible. Similarly, it is not always possible for the supply and demand rate constraints to be met strictly. The scheduler may use a schedule as long as the actual supply and actual demand according to the schedule are within tolerable ranges of the respective specified values *B* and *L*. In contrast, absolute dosage parameters and tolerable supply and demand ranges impose hard constraints: The scheduler considers a failure to meet a *hard constraint* as a non-compliance event that requires a specified action.

2.2 Special Instructions

For each medication M that interacts with other medications or food, the MSS contains a SI part. The part defines additional constraints on separations between doses and changes in dosage parameters of interacting medications. We call interacting medications *interferers*. For each interferer N of M, the separation part for M gives the required temporal separations between each dose of M and any dose of the interferer. We use $\sigma_{min}(N, M)$ to denote the minimum separation to dispense a dose of M after dispensing a dose of N. Typically, $\sigma_{min}(N, M) \neq \sigma_{min}(M, N)$. For example, because Fosamax has to be taken on empty stomach and at least 30 minutes before any food and drink, $\sigma_{min}(Fosamax, Food)$ is 0.5 and $\sigma_{min}(Food, Fosamax)$ is 6.

The *change list part* indicates the required changes, if any, in dosage parameters of M when some interferer affects M. The parameters given by change lists take precedence over dosage parameters of M specified in its DP section as long as the user is taking both M and the interferer. As an example, taking Fosamax in doses of size larger than 10 mg per day with aspirin may increase the likelihood of stomach upset. For this reason, the change list for Fosamax has absolute dose size and separation ranges of [1, 1] and [24, 24], respectively; the flexibility provided by absolute ranges [1, 7] and [24, 168] in the DP part of Fosamax is no longer allowed as long as the user is taking aspirin as well.

We ignore in this paper constraints such as precedence constraint and maximum separation. Precedence constraints restrict the order in which doses of some interacting medications are taken. Maximum separation constraints ensure that some medications are taken sufficiently close together. Our report on a general model of medication scheduling [9] discusses these and other additional constraints.

3 Major Steps and Performance Criteria

Scheduling of interacting medications is done in two major steps. When presented with a MSS, the dispenser scheduler ignores separation constraints between interacting medications in the first step and finds a feasible schedule for every medication M: A *feasible schedule* defines a *usage separation range* $[u_{min}(M), u_{max}(M)]$ which is such that $[u_{min}(M), u_{max}(M)] \subseteq [s_{min}(M), s_{max}(M)]$ of M and there is a dose size in the nominal dose size range of M meeting the supply rate and demand rate of M. The set of feasible schedules selected in this step defines a complete schedule if none of the user's medications has inter-medication separation constraint. The second step is carried out

only when some medications interact and impose inter-medication separation constraints. The OMAT and ODAT algorithms described in Section 4 are used in the second step to take into account these constraints. They are the focus of this paper.

3.1 Consistency and Feasibility

For sake of completeness, we summarize here the work done by the scheduler in the first step: Again, the scheduler ignores in this step the separation constraints listed in the SI part of the MSS. It can use the demand-versus-supply test (DST) [14] described by Figure 2 to check nominal dosage parameters and rate constraints of each medication for consistency and to find a feasible schedule for the medication if such a schedule exists.

Demand-Versus-Supply Test (MD parameters of *M*)
Input: dose-size range [d_{min}, d_{max}], separation range [s_{min}, s_{max}]
supply rate (B, R) and demand rate (L, P) constraints,
duration range [T_{min}, T_{max}]
Output: *feasible* = TRUE;
feasible_schedule = {*NULL*};
1. Check input parameters for validity condition listed below:
   - Valid ranges: $0 \leqq d_{min} \leqq d_{max}$ and $0 < s_{min} \leqq s_{max}$
   - Valid limits: $d_{max} \leqq B$ and $s_{max} \leqq P \leqq T_{min}$
   - Feasible supply rate: $dmin \times \text{Floor}[R/smax] \leqq B$
   - Feasible demand rate: $\text{Ceil}[L/d_{max}] \leqq \text{Floor}[P/s_{min}]$
   - Consistent rate limits: $B/R \geqq L/P$

   if any condition above is not met, *feasible* = FALSE; return;
2. $L/P = 0$, or $B/R = \infty$, return;
3. Carry out steps of one of the following cases:
   a. If $R \geqq P$: find the *minimum_demand_schedule* and the *required_supply* of the schedule;
      if *required_supply* $> B$, *feasible* = FALSE;
      else *feasible_schedule* = *minimum_demand_schedule*;
   b. If $R < P$: find the *maximum_supply_schedule* and the *available_supply* of the schedule;
      if *available_supply* $< L$, *feasible* = FALSE;
      else *feasible_schedule* = *maximum_supply_schedule*;

   return;

Figure 2 Demand-Versus-Supply Test

The constraints given by the MSS for a medication *M* are *consistent* when they do not conflict with each other, i.e., the parameters specifying them meet the necessary conditions checked in Step 1 of DST. Step 1 returns and reports that the MSS is not feasible if the parameters fail to meet any of the necessary conditions. The set of conditions are both necessary and sufficient for several special cases, one of which is that the medication does not have both supply rate constraint and demand rate constraint. This is the reason for Step 2 of the DST test.

In Step 3 of the DST, the scheduler tries to find a feasible periodic schedule that meets the dose

size, separation and rate constraints. We call a schedule according to which k doses are scheduled per period a k-dose schedule. It is a valid k-dose schedule if all dose sizes and separations meet the dose size and separation constraints.

In the case where supply rate interval R is equal to or larger than the demand rate interval P, the test tries to find a feasible periodic schedule called *minimum demand schedule* in Step 3a. This schedule has the longest period and the smallest total dose size among all valid k-dose schedules which are such that $\pi(k) \leq P$ and $\delta(k) \geq L$ where $\pi(k)$ denotes the period of the schedule and $\delta(k)$ denotes the total dose size per period according to the schedule. In essence, a minimum demand schedule meets the demand rate constraint and the total size of doses in any interval no longer than P is as small as possible. The test returns the schedule if the total size of all doses within any interval of length R is no greater than the supply constraint B; otherwise, the test declares failure to find a feasible schedule for the medication under consideration and the MSS infeasible.

Step 3b handles the case when the demand rate interval P is larger than the supply constraint interval R in an analogous way. A *maximum supply schedule* has the smallest period and maximum total dose size per period among all k-dose valid schedules which are such that $\pi(k) \geq R$ and $\delta(k) \leq B$. In other words, the schedule satisfies the supply rate constraint. If the total size of all doses in any interval of length P is at least equal to L, Step 3b returns the schedule; otherwise, it declares the MSS infeasible.

DST is not optimal in the sense that it may declare a MSS infeasible incorrectly. It has been shown in [14] that if Step 3 of DST fails, there is no feasible periodic dosage schedule for the medication. However, we do not know wherever there are non-periodic feasible schedules for the medication.

3.2 Resource Model

For the sake of clarity and to save space, we consider hereafter only those MSS that have feasible schedules for all medications when inter-medication separations are ignored. This way, we can concentrate on the operations and merits of algorithms that deal with additional constraints due to medication interactions. The OMAT and ODAT algorithms described in Section 4 are such algorithms. The rules of these algorithms and operations of the scheduler are based on a resource model according to which the user is the only scarce resource. The scheduler uses two types of virtual resources to manage contention for the user and allocates these entities as if they were

physical processors and resources. There is a processor P_M for every medication M. The scheduler uses P_M to maintain correct separation between doses of M. If M has interferers, then there is also a resource R_M for M. The scheduler uses R_M to maintain the required minimum separations between doses of M and doses of its interferers.

The administration of doses of M are modeled as non-preemptable jobs $J_M(i)$, for i =1, 2,…, on P_M. The execution time of the jobs is in the range [$S_{min}(M)$, $s_{min}(M)$]. Whenever possible, the scheduler uses the minimal nominal separation $s_{min}(M)$ as execution time. It may make the execution time as small as the minimum absolute separation when it is impossible to allocate the job $s_{min}(M)$ units of time.

A job *starts* when the dose it models is released to the user. The first job of each medication M is scheduled to start at some specified time or on best effort basis. In general, the start times of subsequent jobs of M are constrained by two types of deadlines: First, the *inter-stream relative deadline* of $J_M(i)$ with respect to the previous job $J_M(i\text{-}1)$ of M is equal to their maximum absolute separation $S_{max}(M)$. Second, $J_M(i)$ may be required to start by the end of the current demand-rate constraint interval. This requirement imposes an *effective demand-rate deadline* on the job. The absolute start time deadline of $J_M(i)$ is the earliest absolute deadline computed from these relative deadlines. OMAT and ODAT algorithms treat rate constraints as firm constraints; they work only with inter-stream relative deadlines, treating effective demand-rate deadline as infinite.

Some variants of OMAT and ODAT algorithms treat the sequence of jobs of each medication M as a periodic task or sporadic task [9] on P_M. The inter-arrival interval time π_M between consecutive jobs of the task is in the range [$s_{min}(M)$, $s_{max}(M)$]. The task is *periodic* if $s_{min}(M)$ is larger than zero and $s_{max}(M)$ is finite. Otherwise, the task is *sporadic*. The separation in start times of jobs in consecutive periods of a periodic task must be no greater than the relative deadline $S_{max}(M)$. This requirement is equivalent to a jitter constraint. Take as an example a medication with nominal separation range [12, 24] and absolute separation range [8, 28]. We can model the sequence of doses of the medication as periodic task with execution time 12 (or 8) and period 18. The deadline for completing the job in each period is the end of the period. Precise definitions and additional illustrative examples can be found in [9]. Observation 1 follows straightforwardly.

Observation 1 All separation constraints of every medication M are met when jobs modeling their doses are scheduled on P_M as described above.

The scheduler uses the resource R_M in its effort to maintain the required minimum separation between doses of M and doses of its interferers. It follows two rules:

Every job of an interferer N ($\neq M$) of M requires the resource R_M on a shared basis for $\sigma_{min}(N, M)$ units of time beginning from when the job starts. An arbitrary number of jobs of medications other than M can share the resource R_M. The resource is no longer free for the duration when it is thus allocated.

The resource R_M is treated as a permit for jobs of M to start. In other words, a job of M can start at a time instant only if R_M is free at the instant.

Because of rule (2), a job of N can block a job of M for $\sigma_{min}(N, M)$ units of time. We call this time *blocking time* of M by N. The *worst case blocking time* of a medication M is equal to the maximum blocking time of M over all its interferers. The observation below follows as a direct consequence.

Observation 2 Minimum separation constraints specified for all interacting medications are met when resources are allocated to jobs according to Rules (1) and (2).

3.3 Performance Metrics

As we mentioned earlier, the merits of a medication scheduling algorithm are measured from two perspectives: From the perspective of the system whether an algorithm can find a feasible schedule that meets all the constraints specified by the given MSS is important. The parameter we use to measure this dimension of performance is *success rate*. i.e., the chance the algorithm finds a feasible schedule. From the user's perspective, quality of medication schedules is important. We measure schedule quality along two dimensions: adherence to medication directions and user friendliness. These dimensions are in conflict, meaning it is not possible to optimize a schedule along both dimensions.

Adherence to Medication directions Several figures of merit measure how close the schedule adheres to medication directions. They include dose-size variation, separation jitter, and deviation from nominal parameter ranges [9, 10]. In this paper, we measure this aspect of schedule quality by variation and deviation from supply rate and demand rate. As stated earlier, algorithms for scheduling interacting medications cannot always adhere to the rate constraints strictly as they try to satisfy inter-medication separation constraints. This is why OMAT and ODAT families of

algorithms treat supply and demand rates as firm constraints: They temporarily ignore these constraints but check to be sure that the schedules generated by them meet the constraints reasonably closely. Closeness is measured in terms of the actual intake within the specified rate intervals: The *total intake* (or *intake* for short) *of a medication according to a schedule* is the actual total dose size of the medication scheduled over the respective rate interval.

For a given schedule, the *supply rate variation of a medication M* with supply rate (B, R) is the difference between the maximum intake and minimum intake of M within any interval of length R, normalized with respect to the supply B. The *supply rate variation of a schedule* is the maximum supply rate variation over all medications according to the schedule. The *supply rate deviation of a medication M* in MSS according to a schedule is equal to the difference between the maximum intake of M within any interval of length R and the supply B, if the difference is positive; it is equal to zero if the difference is zero or negative. The *supply rate deviation of a schedule* is the maximum supply rate deviation over all medications according to schedule. In essence, the supply rate deviation measures how well the schedule adheres to the supply rate constraint while the supply rate variation measures how widely dose size and separation vary.

Similarly, the *demand rate variation of a medication M* with demand rate (L, P) is the difference between the maximum intake and the minimum intake within any interval length of L. The *demand rate variation of a schedule* is the maximum demand rate variation over all medications according to the schedule. The *demand rate deviation of a medication M* according to a schedule is the difference between the demand L and the minimum total intake in any interval of length P if the difference is positive, and is equal to zero if the difference is zero or negative. The *demand rate deviation of a schedule* is the maximum demand rate deviation over all medications according to the schedule.

User Friendliness We quantify user friendliness by normalized allowed tardiness. The *allowed tardiness* (AT) of a dose k of medication M according to a schedule is the maximum length of time the dose can be delayed without leading to non-compliance. The *normalized allowed tardiness* (NAT) of a dose k of medication M is the ratio of AT and the difference between $s_{max}(M)$ and $s_{min}(M)$. We say an algorithm is *unfriendly* to a dose of the medication if the NAT of the dose is zero, *somewhat friendly* if the NAT is between 0 and 0.3, *friendly* if the NAT is between 0.3 and 0.7, and *very friendly* if the NAT is between 0.7 and 1. In the data presented in Section 5.3, we call the corresponding ranges of NAT unfriendly, somewhat friendly, friendly and very

friendly NAT, respectively.

4 OMAT and ODAT Algorithms

Both OMAT and ODAT families of algorithms make use of priorities. When the scheduler works according to OMAT algorithms, it first assigns priorities to individual medications. It then schedules the medications in priority order, one at the time for the entire duration of each medication. In contrast, when working according to ODAT algorithms, the scheduler assigns priorities to individual doses in individual medications and then schedules doses that are ready to be dispensed in priority order. The scheduler may use a fixed priority scheme that assigns the same priority to all doses in each medication. It may also choose to use a dynamic priority scheme [11], i.e., giving possibly different priorities to individual doses in each medication. Clearly, dynamic-priority scheme is suitable for ODAT algorithms only.

The priority schemes considered here include the ones listed below:

- *MVF* (*most-victimized-first*) scheme gives priorities to tasks, modeling sequences of jobs dispensing doses of medications, based on their worst case blocking times; the longer the worst case blocking time, the higher the priority.
- *MIF* (*most-interactions-first*) scheme gives priorities to tasks based on the numbers of interferers of the corresponding medications; the larger the number, the higher the priority.
- *SSDF* (*shortest-separation-difference-first*) scheme gives priorities to tasks based on the differences between the maximum and minimum nominal separations of the corresponding medications; the shorter the separation difference, the higher the priority.
- *RM* (rate-monotonic) scheme gives priorities to medication tasks based on their periods; the shorter the period, the higher the priority.
- *EDF* (*earliest deadline first*) scheme is a dynamic-priority scheme. It gives priorities to jobs based on their absolute deadlines; the earlier the deadline, the higher the priority.

4.1 Major Data structures

Figure 3 gives data structures used by our scheduling algorithms. The four classes MedicationDirections, DosageParameters, SpecialInstructions and Interferer in Part 1 of Figure 3 are used to record direction and special instruction parameters. The scheduler creates an instance of each class for every medication (say *M*) in a MSS and initializes the variables in it according to MSS. Specifically, elements of the instance of DosageParameters for medication *M* hold values

delimiting the ranges of duration (t), nominal dose size (nd), nominal separation (ns), absolute dose size (ad), and absolute separation (as), as well as supply rate parameters (b, r) and demand rate parameters (l, p) of the medication. The elements min_to_interferer and min_from_interferer of the instance of Interferer created for an interferer N of the medication M hold values of minimum separations $\sigma_{min}(M, N)$ and $\sigma_{min}(N, M)$, respectively.

Data Structures of OMAT and ODAT algorithms

Part 1: Data structures for recording the information of each medication in a MSS

```
Class MedicationDirections{
    int med_id;    DosageParameters dp;    SpecialInstructions si;
}
Class DosageParameters{
    int t_min, t_max;
    int ns_min, ns_max, as_min, as_max;
    int nd_min, nd_max, ad_min, ad_max;
    int b, r, l, p;
}
Class SpecialInstructions{
   List<Interferer> drug_and_food_interaction;
   List<DosageParameters> change_lists;
}
Class Interferer{
  int med_id;   int min_to_interferer;   int min_fr_interferer;
}
```

Part 2: Data structures of resource model for each medication in a MSS

```
Class ResourceModel{
  int[] resource = new int[t_min];   int[] processor = new int[t_min];
  bool feasible = TRUE;   List<int> schedule = NULL;
  int priority = 0;
}
Class JobModel{
  int  release_time = 0;   int execution_time = ns_min;
  int deadline = release_time + execution_time;
}
```

Part 3: Data structure of priority schemes

```
enum PrioritySchemes{ RM, MVF, MIF, SSDF, EDF};
```

Figure 3 Data structures of scheduling algorithms

Class ResourceModel in Part 2 is used to implement the resource model described in Section 3.2. resource and processor are both integer arrays of size equal to the minimum duration over which the medication is administered. The values of resource[k] and processor[k] for M tell the scheduler whether R_M and P_M, respectively, are free or allocated (indicated by values 0 and 1, respectively) at time k since the beginning of the schedule (i.e., since time 0).

Medication M is feasible if the scheduler can find a valid start time, satisfying all hard constraints, for every job of the medication task M. Whether M is feasible is indicated by the value of the variable feasible in ResourceModel; the value is TRUE if M is feasible and is FALSE

otherwise. schedule is the head of a sorted linked list of integers; each integer in the list gives the start time of a job of *M*. priority is an integer variable whose value represents the priority of the medication: The smaller the value, the higher the priority. The value of priority for *M* is chosen according to PrioritySchemes listed in Part 3 in Figure 3.

The class JobModel gives the real-time workload parameters of the jobs in the medication task *M*. The value of release_time is the release time of a job $J_M(i)$, i.e., the instant of time when job is ready for execution. We consider here only the case where the release time of the first job $J_M(0)$ is at the beginning of the schedule for every medication. The release time of each job $J_M(i)$ for $i > 0$ is the time when $J_M(i-1)$ completes. As it will become evident shortly, extension of the algorithms to handle the general case where the first jobs are released at arbitrary times is straightforward. execution_time of *M* is the amount of time jobs of task *M* occupies P_M once the job starts. execution_time is the range delimited by the minimum absolute separation and the minimum nominal separation of *M*. The initial value of this variable is the minimum nominal separation because the scheduler uses this value usually. It uses values a smaller value in the allowed range only when it cannot find a feasible schedule of *M* without doing so. deadline of $J_M(i)$ here is the inter-stream relative deadline of the job.

4.2 Scheduling Operations

Figures 4 and 5 give pseudo-code descriptions of basic versions of OMAT and ODAT algorithms, respectively.

The inputs of the algorithms are MedicationDirections of all medications in a MSS and PrioritySchemes used by the algorithms. As output, an algorithm returns the value of the Boolean variable feasible, which indicates whether the MSS is feasible. The feasible_schedule [ ] array of schedule lists, also returned as output, contains a schedule list head (hence, the schedule) for each of the medications in the MSS if the algorithm finds the specification feasible.

The first step of basic OMAT and ODAT algorithms is to create an instance of ResourceModel and an instance of JobModel for each medication in MSS. The variables within the instances are initialized as shown.

Major Steps of Basic OMAT Algorithms Using a basic OMAT algorithm, the scheduler assigns a priority to each medication according to PrioritySchemes in Step 2. The bulk of the work is done in Step 3.

```
One-Medication-At-A-Time
Input: List<MedicationDirections> MSS, PrioritySchemes;
Output : feasible = TRUE;
         feasible_schedule [number_medications] = {NULL};
1. For every medication listed in MSS, create an instance of
   JobModel, ResourceModel and list head schedule.
2. Assign priority to each medication according to PrioritySchemes.
3. For each medication Mi from the one with the highest priority to
   lowest, do the following:
     latest_start_time = 0; current_medication_feasible = TRUE;
     do while ( current_medication_feasible == TRUE ) {
          A. Call FindAvailableTime ( ) to get the earliest available
             time instant x
          B. if x is found,
               if ( x >= t_min of Mi ) break;
               Insert x into the start time list schedule of Mi
               for ( x <= k < x + ns_min of Mi )
                    processor[k] =1;
               for every interferer N of Mi listed in MSS
                   for(x – min_to_interferer N <= k < x +
                     min_fr_interferer N ) set resource[k] of N to 1;
               latest_start_time = x;
          C. else x is not found
               current_medication_feasible = FALSE;
               feasible = FALSE;
     }
     if feasible == TRUE, feasible_schedule[Mi] = schedule of Mi
4. Return feasible and feasible_schedule[number_medications]
```

Figure 4 Basic OMAT algorithms

```
One-Dose-At-A-Time
Input: List<MedicationDirections> MSS, PrioritySchemes;
Output : feasible = TRUE;
         feasible_schedule [number_medications]= {NULL};
1. For every medication listed in MSS, create an instance of
   JobModel, ResourceModel and list head schedule.
2. schedule_complete = 0;
   release_time_current_jobs[number_medications] = {0};
   While (feasible == TRUE) and
         (schedule_complete < number_medications){
     A. Determine based on release_time_current_jobs[ ] the current
        job j with the earliest release time and highest priority among
        current jobs of all medications , say j is of medication Mi
     B. do
          a. Call FindAvailableTime ( ) to get the earliest available
             time instant x.
          b. if x is found,
               if ( x >= t_min of Mi ) schedule_complete++ break;
               Insert x into the start time list schedule of Mi
               for ( x <= k < x + ns_min of Mi ) processor[k] =1;
               for every interferer N of Mi listed in MSS
                   for ( x – min_to_interferer N <= k < x +
                     min_fr_interferer N ) set resource[k] of N to 1;
               release_time of Mi's current job = x + ns_min.
          c. else x is not found
               feasible = FALSE.
     }
3. Return feasible and feasible_schedule[number_medications].
```

Figure 5 Basic ODAT algorithms

In Step 3, the scheduler considers medications in priority order one at a time: For each medication M, the scheduler schedules jobs (i.e., doses) of M one at time, starting from the first job until it finds either M is not feasible or the possible start time of the current job (i.e., the job being scheduled) is larger than the minimum duration of M. For the current job, the scheduler calls FindAvailableTime( ) shown in Figure 6 to find the *earliest available time instant x,* which is the earliest time instant between the release time and deadline of the job at which both R_M and P_M are free. If an instant x is found, the scheduler schedules the job to start at x, and adds x to the start time list schedule of M. When the job starts, it occupies P_M for execution_time units of time; the scheduler updates the processor array of M accordingly. The job does not actually hold R_M as the allocation rule described in Section 3.2 requires only that the resource is free at x. However, if M has interferers, the scheduler marks resource array of each interferer N are allocated according to rules of the resource model. (We note that the pseudo code also states that time units of resource array of N from min_to_interferer of N to x are also marked as occupied. This is done to simplify FindAvailableTime( ) function.) The scheduler then moves on to schedule the next job in M. If the scheduler successfully schedules all the jobs within the duration of M, it set the element in feasible_schedule [ ] for M to the list head schedule.

A failure of FindAvailableTime( ) to return an earliest available time indicates that the

medication *M* is not feasible. The variable feasible is set to FALSE. The value of this variable indicates whether that the basic version of OMAT algorithm has succeeded or failed after all medications in a MSS are scheduled. Again, the array feasible_schedule [ ] provides the heads of schedule lists of all the medications if the algorithm succeeded.

```
FindAvailableTime( )
Input: JobModel and  ResourceModel
Output: feasible and available time x
1.   From the time of the job is released to the job's deadline,
       a.   if there is a time r such that resource[i] == 0 and processor[i]
            == 0, x =  r ; feasible = TRUE;
       b.   else feasible = FALSE;
2.   Return feasible and x.
```

Figure 6 FindAvailableTime function

Major Steps of ODAT Algorithms When the scheduler uses an ODAT algorithm, the scheduler assigns priorities to jobs according to PrioritySchemes when the jobs are released. It schedules released jobs in priority order. Step 2A in Figure 5 is a way to implement this strategy efficiently: The scheduler uses the array release_times_current_jobs [ ] to keep track when the current job(s) of some medication(s) is released and ready to be scheduled, rather than checking each time instant for released current job(s).

We say that the schedule of a medication *M* is *complete* when the possible start time of the current job of *M* is later than the minimum duration of *M.* The scheduler continues to schedule jobs as they are released in priority order until either it fails to find a possible start time for the job currently being scheduled or the schedules of all medications in the given MSS are complete. This work is done in Step 2. Step 2B is similar to Step 3 of OMAT algorithms.

Advanced OMAT Algorithms We will see in the next section that in terms of success rate, basic ODAT algorithms outperform basic OMAT algorithms consistently. By examining sample schedules generated by basic OMAT algorithms in simulation experiments, we found that doses of high priority medications are scheduled either as soon as they are released or shortly afterward. Consequently, doses of higher priority medications are scheduled close together. Resources of interfering medications of lower priorities are more likely to be occupied when the scheduler tries to find available start times for them. This is usually why the basic OMAT algorithms fail to schedule medications of lower priorities for all priority schemes.

The *advance OMAT* algorithm is designed to solve this problem. Unlike the basic algorithm,

the advanced OMAT algorithm schedules doses of medications as close to their respective deadlines as possible. As data on success rate presented in the next section show, with this enhancement, one medication at a time indeed is a better strategy than one dose at a time.

5 Performance Evaluation

We conducted several simulation experiments to determine how well basic OMAT and ODAT algorithms perform for a variety of sample MSS's and how their performance depend on the choice of priority scheme. We also wanted to experiment with a way to enhance the algorithms and determine its effectiveness. This section first describes how sample MSS's were generated. It then describes data on success rate and schedule quality.

5.1 Simulation Experiments

The sample medication schedule specifications (MSS) used in our simulation experiments is characterized by several parameters. They include number of medications, dose frequency, interference probability and interference severity. For each simulation run, we generated 1,000 MSS samples. Sample data collected during all runs indicated that this number is large enough to yield 95 percent confidence.

Number of Medications While experimenting with the number n of medications listed in MSS, we found that specifications (i.e., MSS) with fewer than 3 medications are almost always feasible. In contrast, the specifications with more than 20 medications are almost always not feasible. For this reason, we narrow the range of this parameter to [3, 20].

In all experiments, we started with $n = 3$. We then increased the value of this parameter by one per step until $n = 20$.

Dose Frequency In subsequent discussion, we divide medications into three categories depending on how frequent they are taken. The categories are frequent, typical, and infrequent. A medication is in the *frequent* category if it is taken more than four times a day. A medication in the *typical* category is taken one to four times a day. As its name implies, a medication in the *infrequent* category is taken at most once per day and may even be as infrequently as once a month.

For the same reason, we generated separation constraint parameters of medications in MSS samples from different probability distributions for different categories of medication. Nominal and absolute separations and values delimiting their ranges are related as described below. The

even distributions from which sample values were selected are listed in Table 1. The times delimiting the ranges are in minutes.

Table 1 Distributions of separation parameters

	Frequent	Typical	Infrequent
s_{min}	[1, 360]	[360, 1440]	[1440, 43200]
NSRW	[1, 120]	[120, 480]	[480, 14400]
DNAMIS	[1, 120]	[1, 120]	[480, 14400]
DNAMAS	[1, 120]	[120, 480]	[480, 14400]

- The minimum nominal separation s_{min} for each sample medication was selected independently from the specified distribution.
- The maximum nominal separation s_{max} for the sample medication equals to s_{min} + NSRW. The parameter NSRW (nominal separation range width) was selected from the distribution listed in the table.
- The minimum absolute separation S_{min} is equal to max (s_{min} – DNAMIS, 0). DNAMIS (difference between nominal and absolute minimum separations) was selected independently from the specified distributions.
- The maximum absolute separation S_{max} is equal to s_{max} + DNAMAS where DNAMAS (difference between absolute and nominal maximum separations) was selected independently from the specified even distribution.

Interference Probability Each sample medication in a MSS interferes with other medications with *interference probability* ρ. That interference probabilities of all medications are identical is a reasonably accurate model of medication interactions.

In the experiments described below, we begin with $\rho = 0.1$ (i.e., medications rarely interfere). We then increase this parameter by 0.2 per step until $\rho = 1$ (i.e., medications always interfere).

Interference Severity For a medication M that interferes with another medication N, we capture how severe the effect of interference by *interference severity* $\delta(M, N)$. Given the minimum inter-medication separations $\sigma_{min}(M, N)$ and $\sigma_{min}(N, M)$ of the medications, $\delta(M, N)$ is the ratio

$$\delta(M, N) = [\sigma_{min}(M, N) + \sigma_{min}(N, M)] / \max (S_{max}(M), S_{max}(N))$$

It is easy to see that the larger this ratio, the more constrained the schedule. Indeed, the scheduler cannot maintain separations between doses of M and N without violating the maximum

absolute separation of one or both medications when this ratio is larger than 1. So, we initialized this parameter for each pair of interfering medications M and N at 0.1, and then increased it by 0.2 per step until the ratio becomes 1.

Dose Size and Rate Constraint Parameters To increase the chance that individual sample medications are consistent and feasible, we first generated sample values of their rate constraint parameters. We then used the DST algorithm described in Figure 2 to find a feasible dose size of the medication and use the dose throughout the schedule. This is consistent with the major steps of scheduling described earlier in Section 3.1.

Specifically, we first selected for each sample medication the total intake parameters B and L independently from the even distribution [1, 100]. Rate intervals R and P were independently selected from the even distribution [S_{max}, 86400]. The samples failing the consistency test of DST or failing to yield a feasible dose size were thrown away.

5.2 Data on Success Rate

Data collection on success rates in a series of experiments are shown in Figures 7-11. They show how the rates depend on medication types, number of medications and priority scheme and interference probability and severity.

Dependency on Medication Type In order to keep the number of parameter combinations from being overwhelming large, one of the first experiments we did was aimed at determining how easy or difficult different categories of medications are to schedule. For this experiment, sample medications selected for each simulation run are of the same category. Figure 7 summarizes the data collected from the simulation run where the priority scheme used by algorithms is MVF and the number of medications is 5. From the data, we can see that dose frequency has noticeable effect on success rate. The success rates of all OMAT and ODAT algorithms are higher for sample medications with lower dose frequency. Since lower dose frequency means larger separations between doses, this behavior is an expected one.

Dependency on Number of Medications and Priority Scheme To determine how success rate depends on number of medications, we generated MSS with medications of mixed category. Specifically, we selected the nominal minimum separation s_{min} for each sample medication from the even distribution [1, 14400]. We then categorized the sample and selected the other constraint parameters of the sample in relationship to s_{min} according to Table 1. ρ and δ were selected

independently from even distribution over the range [0.1, 1].

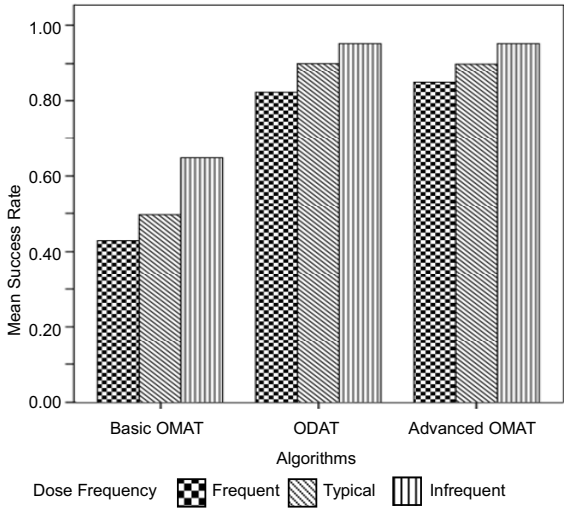

Figure 7 Success rates of different medication categories

We compare the success rates achieved by our algorithms when the scheduler works with strictly smaller nominal separation ranges of all medications with the success rates achieved when the scheduler works with longer absolute separation ranges. Data on success rates for the two cases are shown in Figure 8 and Figure 9, respectively. We can see from the data that by taking advantage of the more relaxed timing constraints, the scheduler can successfully find schedules in many cases where it would fail if it were restricted by the more stringent constraints. Unfortunately, as we will see later in this section, the gain in success rate is obtained at the expense of user friendliness. The schedules that meet the absolute separation constraints but not nominal separation constraints are less tolerant to user tardiness.

Specially, the number *n* of medications in a MSS in parts of Figure 8 and Figure 9 equal to 3, 5, 10, and 15. These figures show that as number *n* of medications increases, success rates decrease for all algorithms. This is especially true for basic OMAT algorithms. When number *n* of medications is more than 10, success rates of finding schedules meeting nominal separation constraints are almost 0, except for advanced OMAT algorithms. Success rates are more than 60% if we accept schedules that may violate nominal separation constraints but meet all the absolute separation constraints. Both Figure 8 and Figure 9 show that advanced OMAT algorithms have higher success rates than the corresponding algorithms in the other two families.

Figure 8 shows clearly that SSDF priority scheme has the highest success rates compared with other priority schemes for all algorithm families. In this case, available start times for jobs of medications are decided by their nominal separations. The smaller the difference between nominal separations, the harder it is to find available start times for jobs of the medication. It makes sense to give medications with smaller separation differences higher priorities. The data indeed validate the common sense strategy. Figure 9 does not show SSDF scheme the clear winner. Rather, the figure points out that the MVF priority scheme consistently performs well, especially when the number of medications in a MSS sample is large. It is harder to find available start times for jobs (i.e., doses) of medications whose resources are occupied by longer jobs of their interferers. Compared with other priority schemes, MVF priority scheme keeps the fraction of time when their resources are occupied small.

Compared with OMAT algorithm families, the differences in success rates of ODAT algorithms using different priority schemes are relatively small. The reason is that doses of different medications are seldom ready to be scheduled at the same time. Hence, the order in which doses are scheduled according to an ODAT algorithm is often determined by their release time.

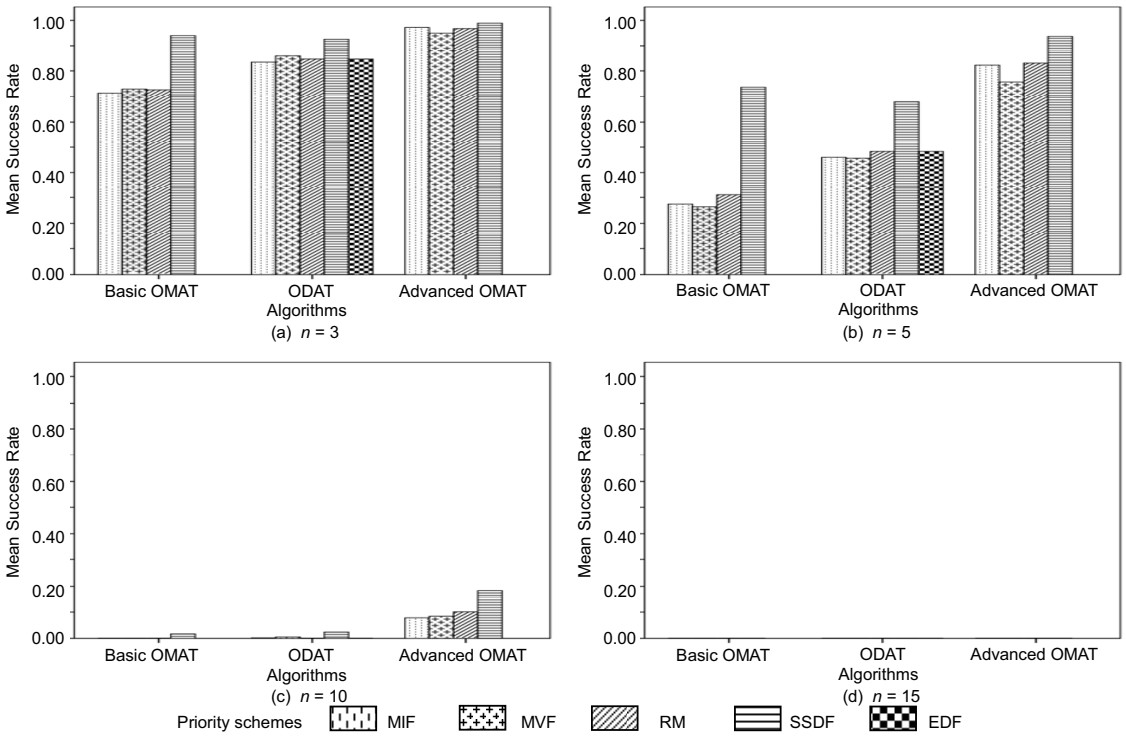

Figure 8 Success rates of schedules meeting nominal separation constraints

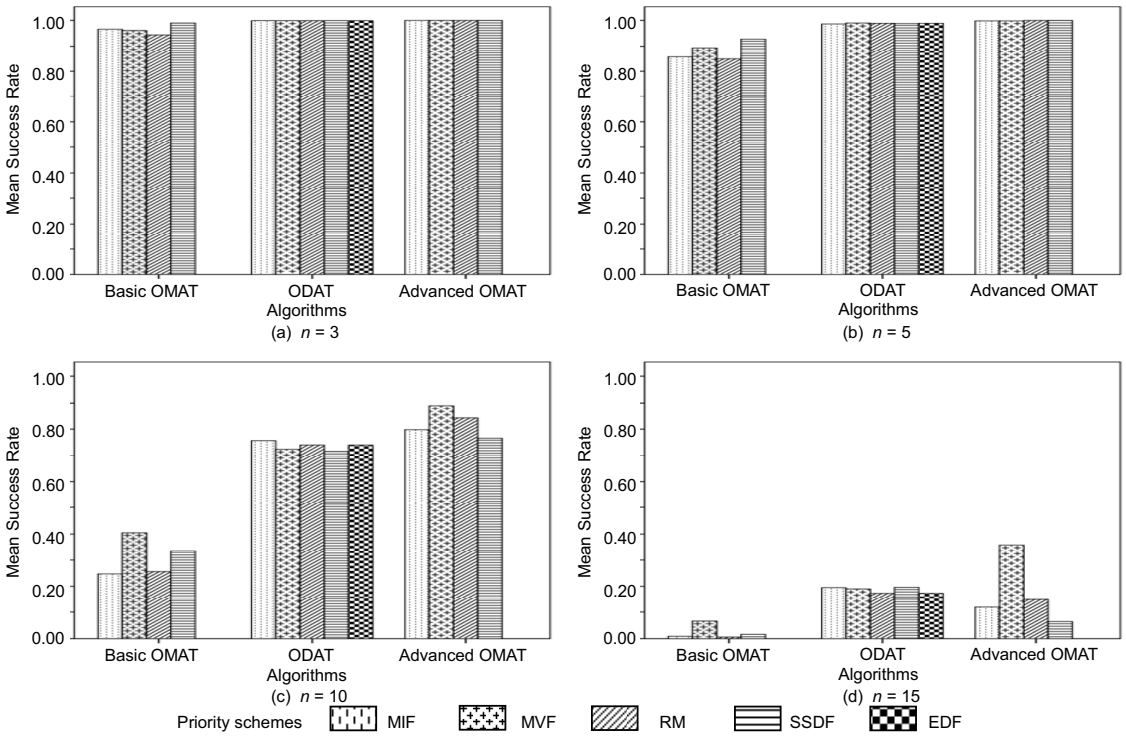

Figure 9 Success rates of schedules meeting absolute separation constraints

Dependency on Interference Probability and Severity To determine how success rate depends on interference probability and severity, we used the same MSS samples as the ones used for Figure 8 and Figure 9 but gave ρ and δ different values. Specifically, the values of these parameters were initialized at 0.1, and then increased independently by 0.2 per step until the values become 1. In this and subsequent experiments, we allows the scheduler to use the more relax constraints imposed by absolute separations, except where it is stated otherwise. The advanced OMAT algorithm with MVF priority scheme was used here because of it achieved the highest success rate amongst all priority schemes considered here.

Figure 10 shows that when the values of ρ or δ are 0.1, success rates for different numbers of medications in a MSS are almost 90%. In this case, medications rarely interfere with each other. When the values of ρ and δ are both 0.3, success rates are still more than 75%. Even when ρ and δ are 0.3, resources of individual medications are free most of the time, allowing their own jobs to start most of the time.

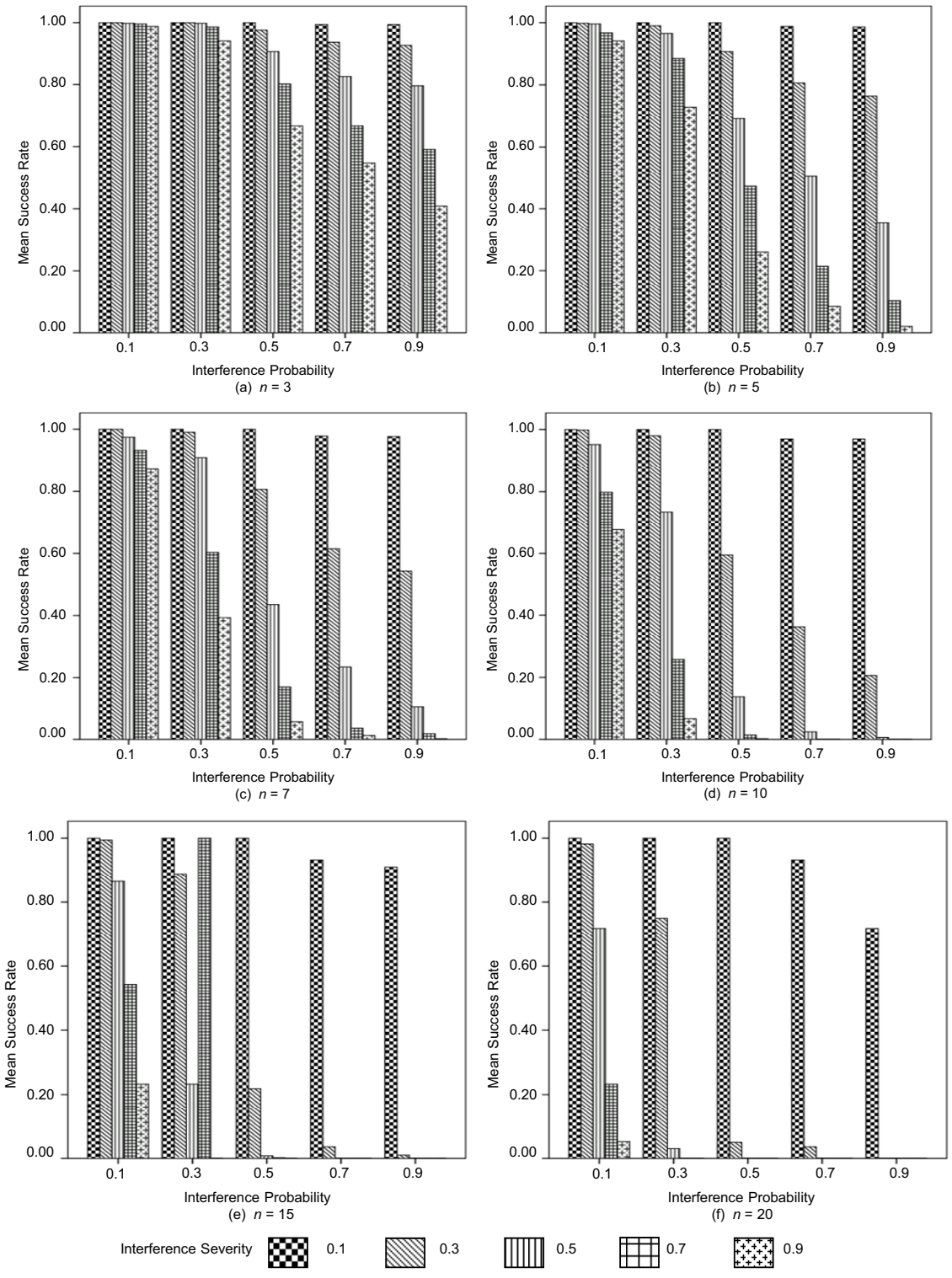

Figure 10 Success rate of advanced OMAT_MVF algorithm as function of interference parameters

However, as part (f) of Figure 10 shows that when the values of ρ or δ are larger than 0.5, the success rate is almost zero for all cases except when *n* is small. For example, we can see from parts (c), (d) and (e) that when the values of ρ and δ are both 0.5, success rate is only 50% when *n* = 7. The success rate decreases to almost 0 when *n* becomes 10 or larger. This is because as ρ increases resources of individual medications are more and more likely to be allocated and are allocated for longer and longer time as δ increases. The effects of these increases becomes more serious when *n* is larger. Part (b) of Figure 10 supports this explanation: It shows that as long as one of ρ and δ is less than 0.5 and the other is less than 0.7, the success rate with *n* = 5 can more than 50%. When *n* = 3, the success rate is still more than 40%, even when ρ and δ are both 0.9.

5.3 Data on Schedule Quality

As stated in Section 3.3, we measure the quality of schedules by four criteria: normalized allowed tardiness (NAT), variations of supply rates and demand rates, deviations of supply rates and demand rates. Results are shown in Figures 11-22.

Figures 11-14 show the percentage of doses with different ranges of NAT categories achieved by scheduling algorithms. The MSS samples used in this experiment are generated with same way as the ones used to produce the data shown in Figures 8 and 9. We can see that the percentages of doses to which the advanced OMAT algorithms are friendly and very friendly are consistently larger than the percentages of doses with NAT in these ranges achieved by algorithms in other families. In short, advanced OMAT algorithms are more user friendly, largely because they schedules doses later. On the other hand, priority schemes do not affect the performance in this respect. Moreover, whether the scheduler works with smaller nominal separation ranges or larger absolute separation ranges makes no noticeable difference. Finally, the percentage of doses with NAT equal to zero increases with *n* as expected.

Figure 15 and Figure 16 show the mean supply rate deviations achieved by the scheduling algorithms for number *n* of medications in a MSS equal to 3, 5, 10 and 15, respectively. We can see that ODAT algorithms have the highest supply rate deviation. Analysis of sample schedules show that ODAT algorithms schedule doses at a higher frequency than algorithms in OMAT families. Consequently, the total intake for each medication scheduled by ODAT algorithms is larger. Advanced OMAT algorithms have the lowest supply rate deviations. Because these algorithms schedule medications as late as possible, the total intake of each medication

scheduled by advanced OMAT algorithms are less than the other two algorithms. Figure 17 and Figure 18 show the mean demand rate deviations achieved by the scheduling algorithms for number *n* of medications in a MSS equal to 3, 5, 10 and 15, respectively. For the same reasons stated above, ODAT algorithms have the lowest demand rate deviations and advanced OMAT algorithms have the highest demand rate deviations among the algorithms.

Figure 19 and Figure 20 show the mean supply rate variations achieved by scheduling algorithms, and Figure 21 and Figure 22 show the mean demand rate variations achieved by scheduling algorithms, for number *n* of medications in a MSS equal to 3, 5, 10 and 15. Compared with other algorithms, advanced OMAT algorithms have the lowest values in both supply rate variation and demand rate variation. This is because the schedules achieved by advanced OMAT algorithms are close to be periodic. Basic OMAT algorithms have the higher variations in both supply rate and demand rate because the resources and processors are likely occupied and it is harder to make the schedule periodic. Therefore, the total intakes of a medication in different time slots vary more widely.

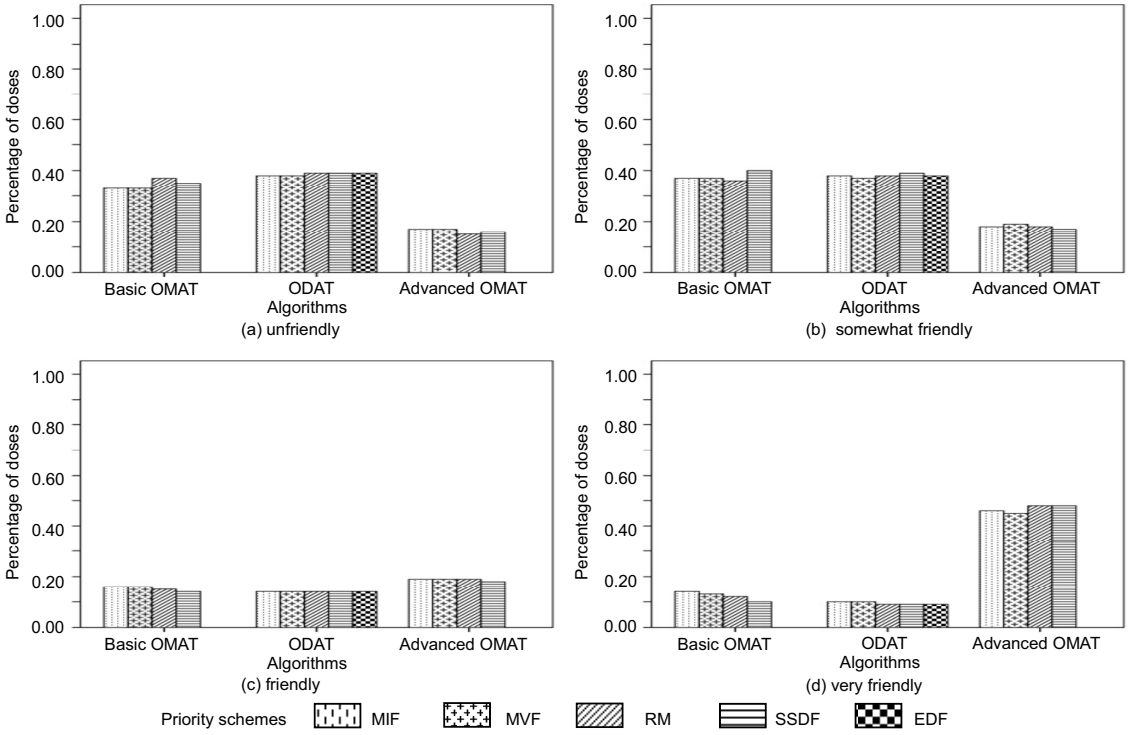

Figure 11 NAT of schedules meeting nominal separation constraints when n = 3

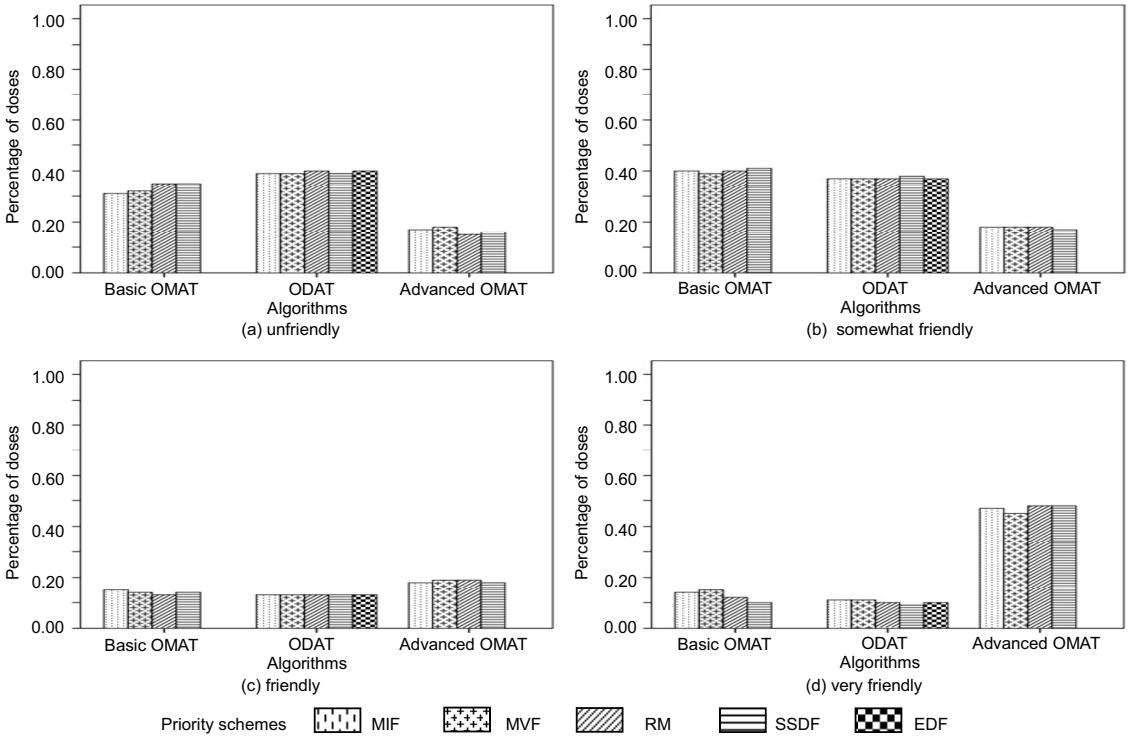

Figure 12 NAT of schedules meeting absolute separation constraints when n = 3

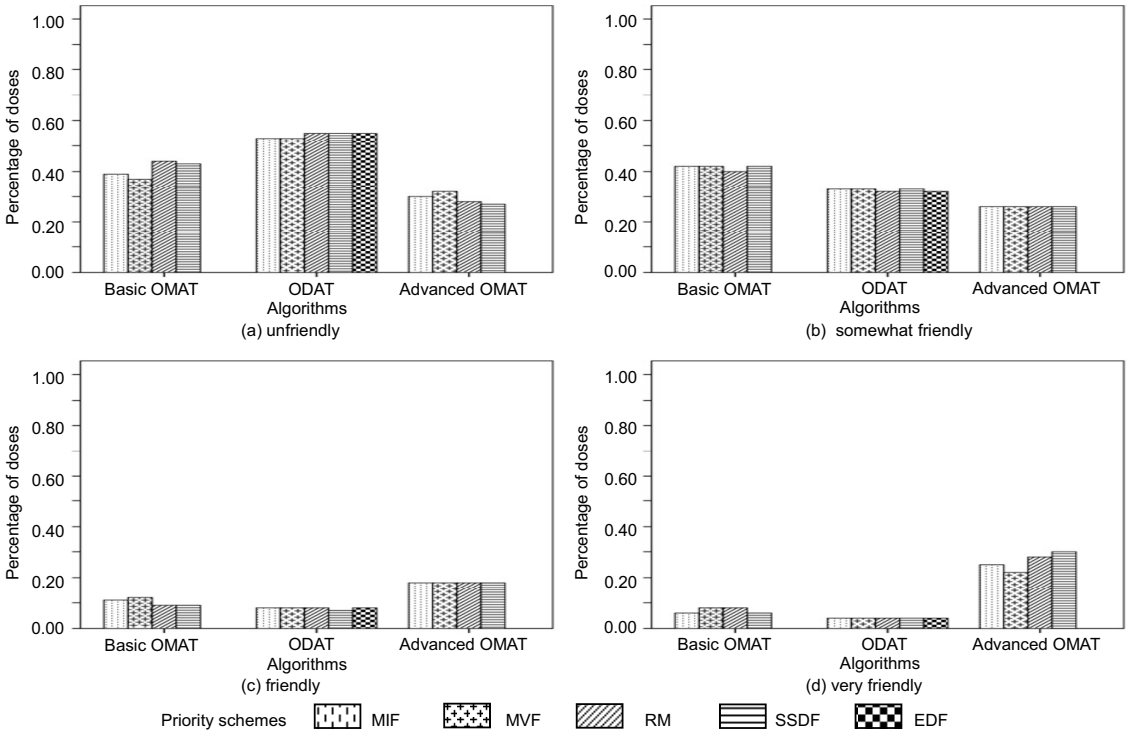

Figure 13 NAT of schedules meeting nominal separation constraints when n = 5

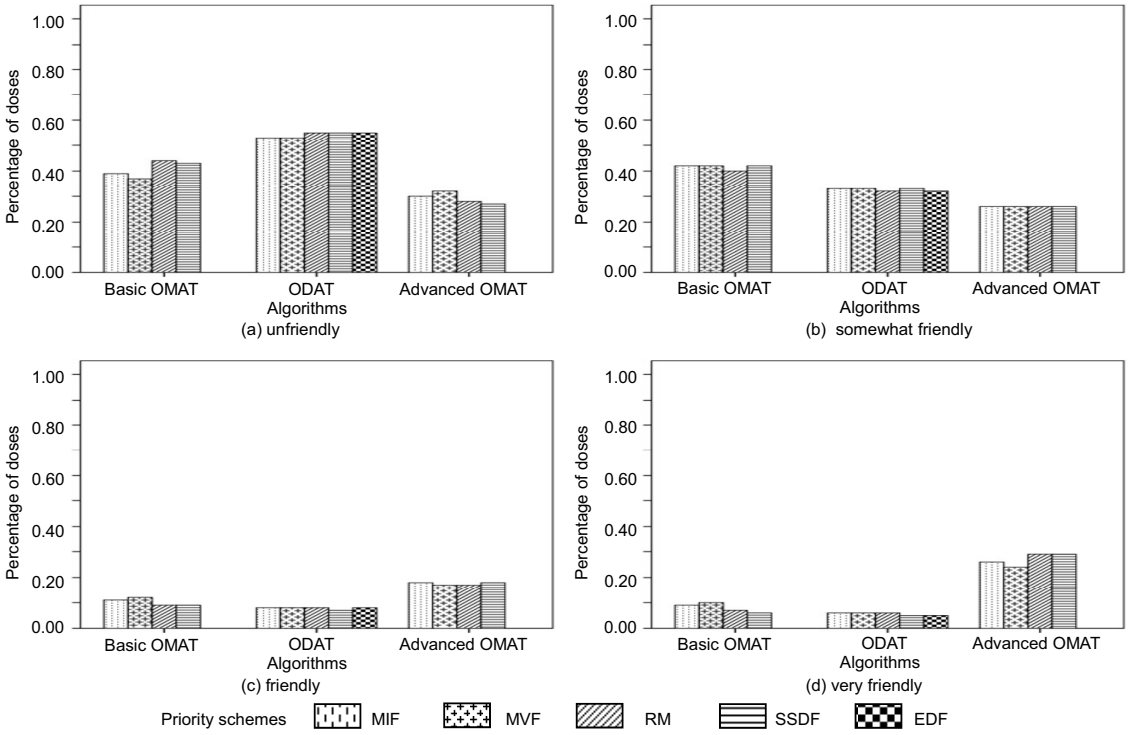

Figure 14 NAT of schedules meeting absolute separation constraints when n = 5

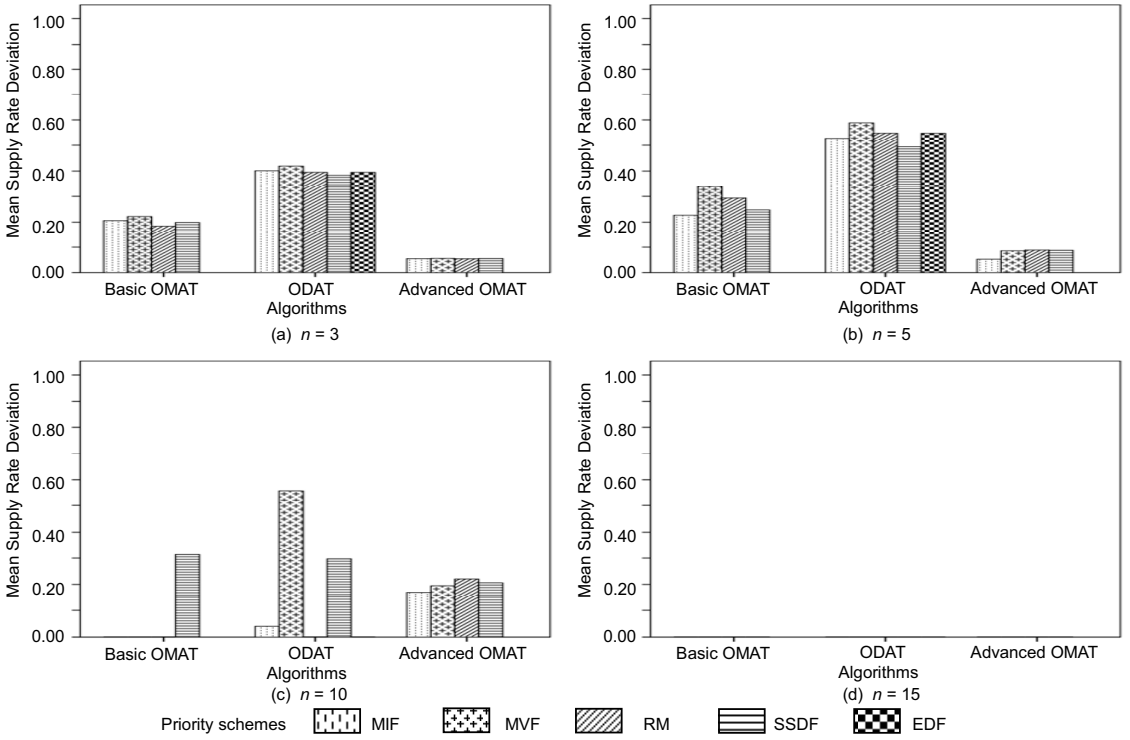

Figure 15 Supply rate deviations of schedules meeting nominal separation constraints

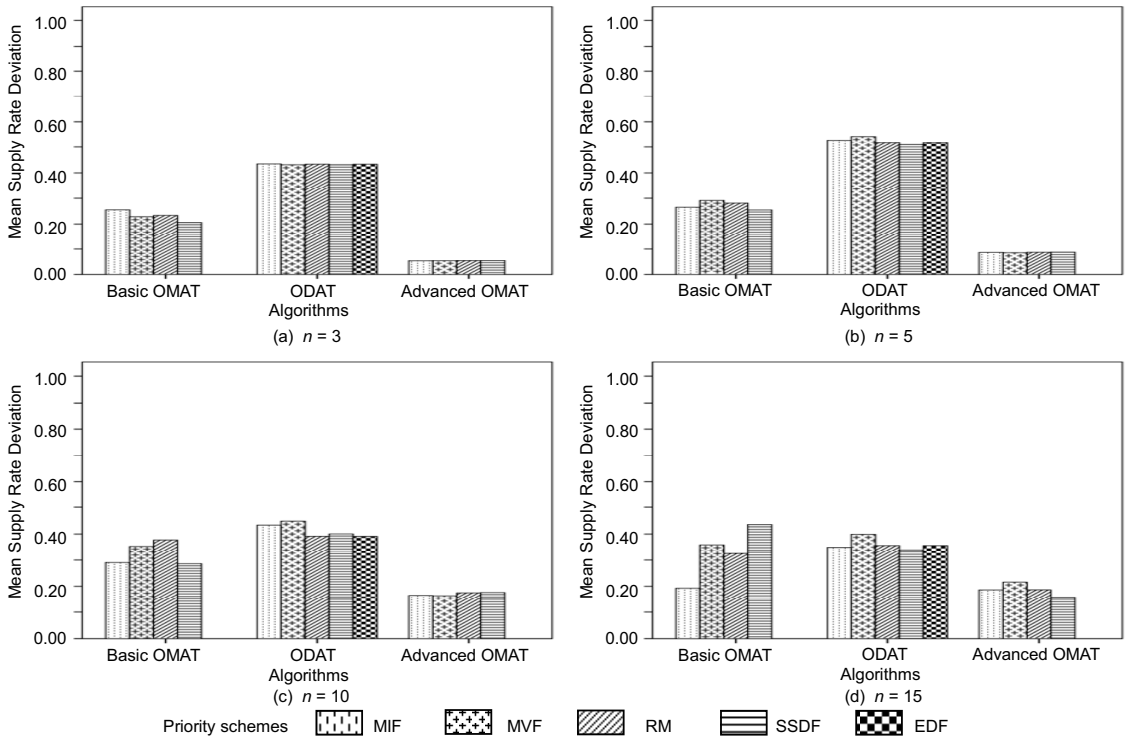

Figure 16 Supply rate deviations of schedules meeting absolute separation constraints

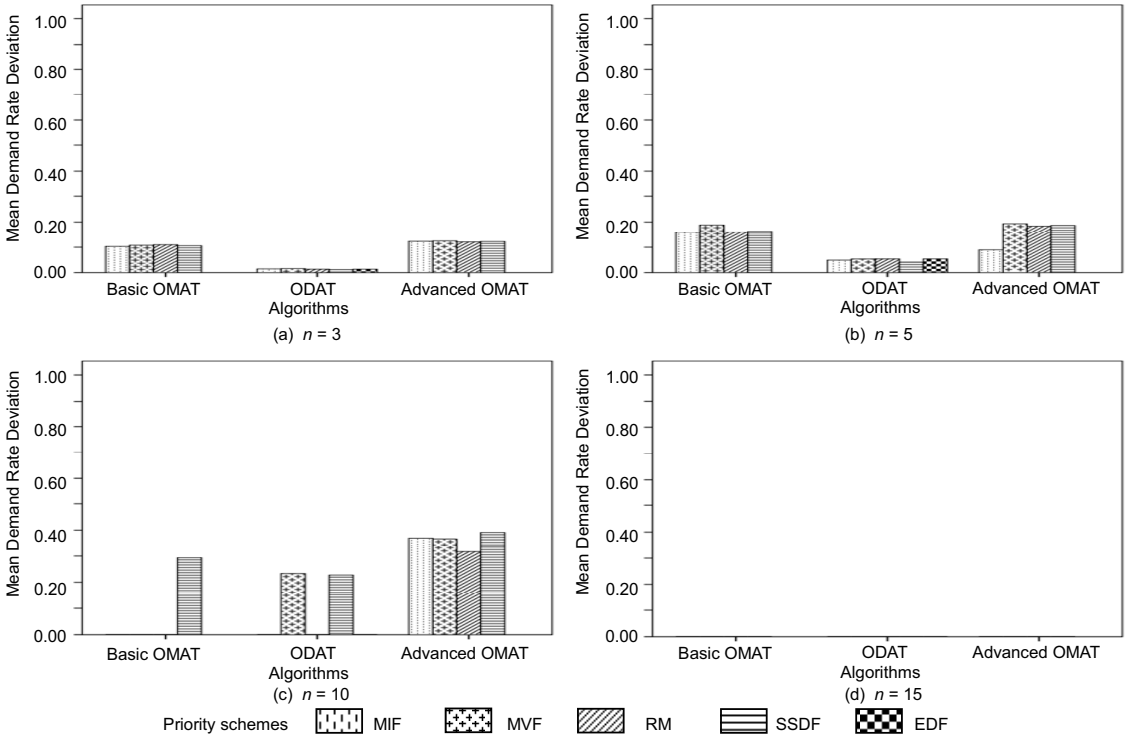

Figure 17 Demand rate deviations of schedules meeting nominal separation constraints

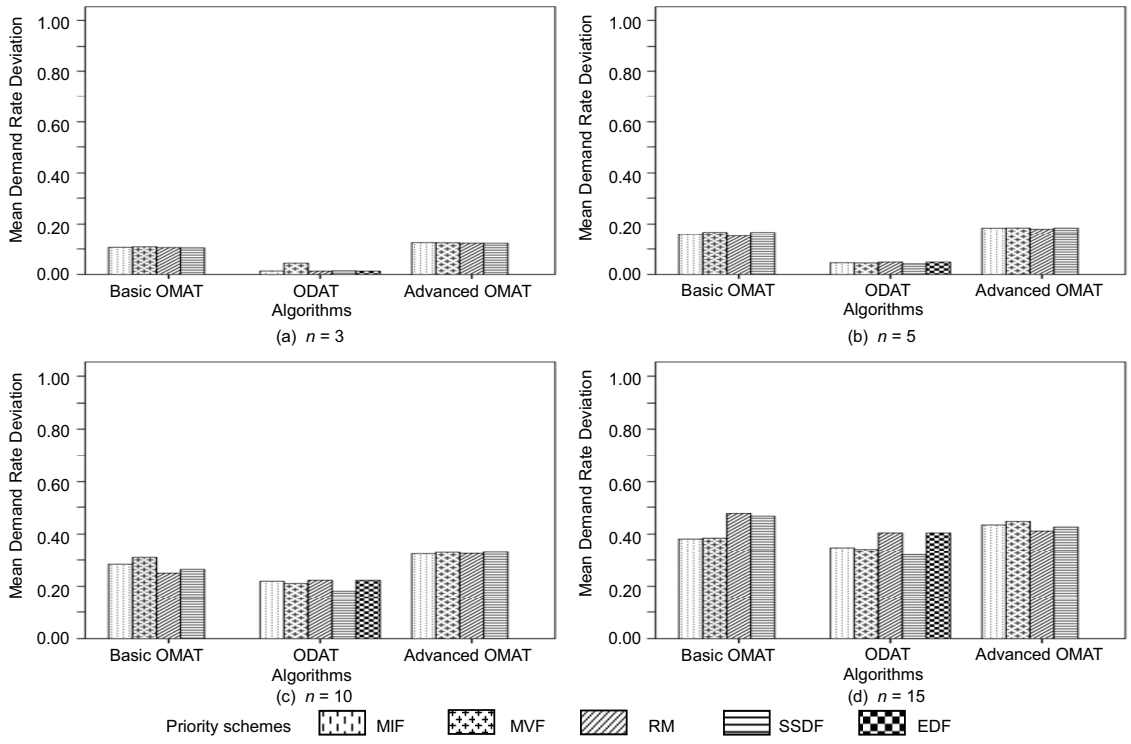

Figure 18 Demand rate deviations of schedules meeting absolute separation constraints

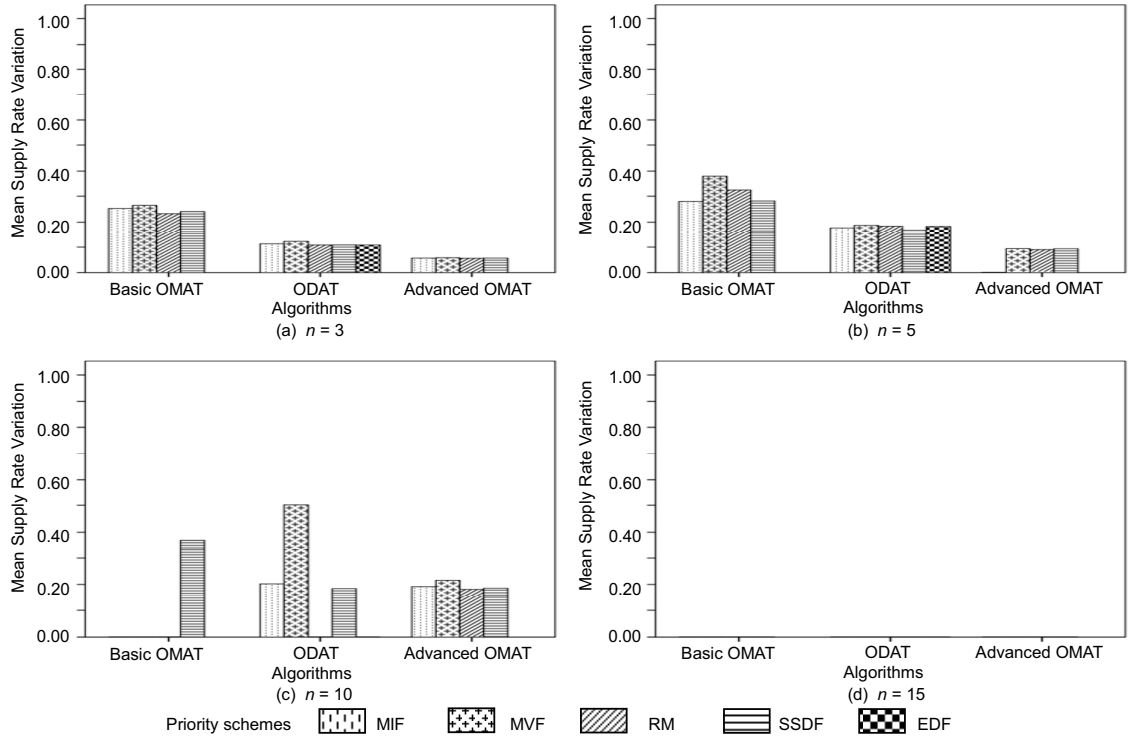

Figure 19 Supply rate variations of schedules meeting nominal separation constraints

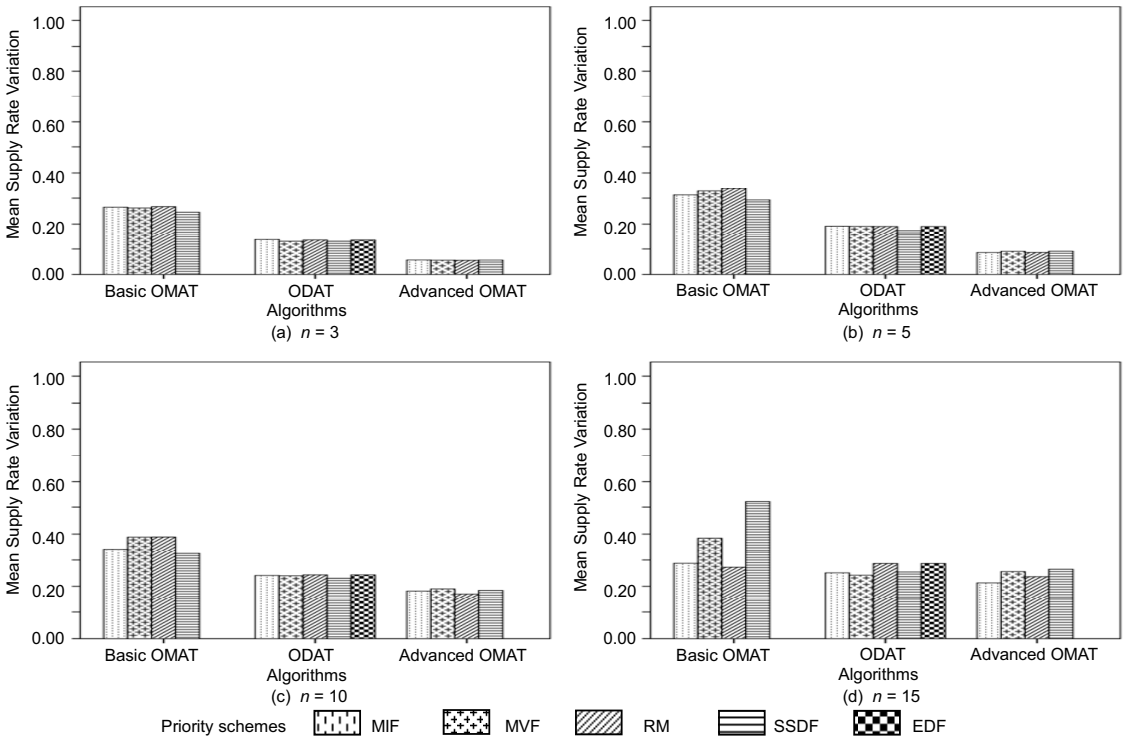

Figure 20 Supply rate variations of schedules meeting absolute separation constraints

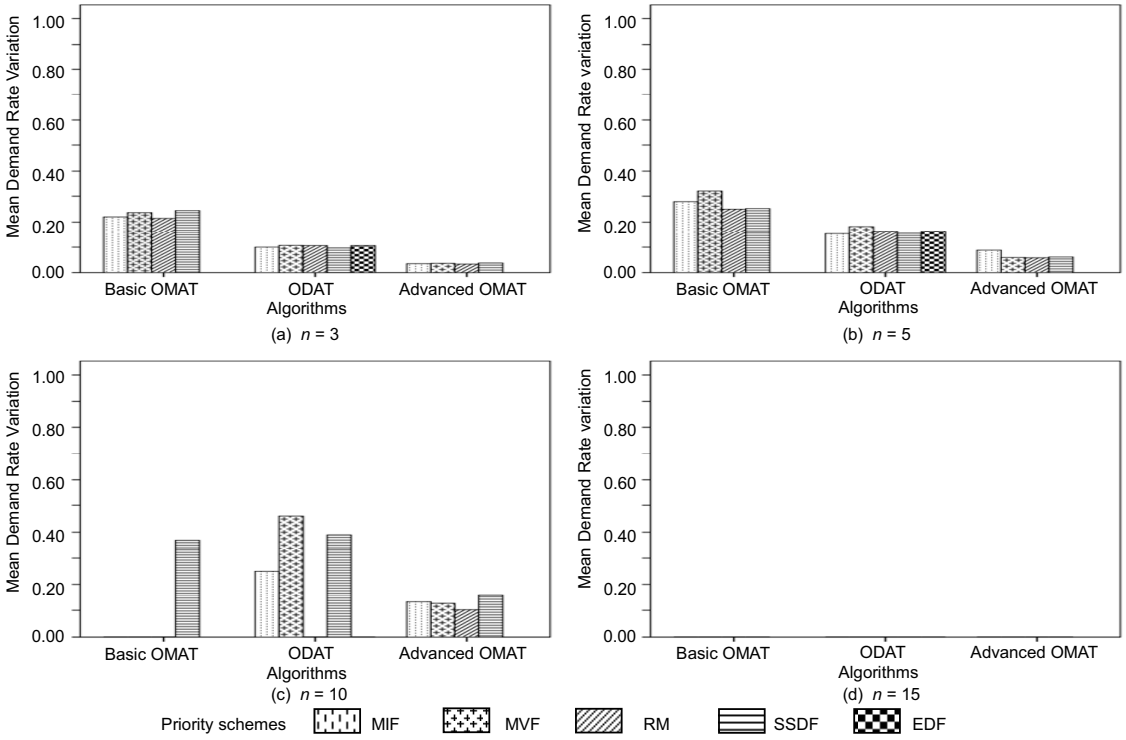

Figure 21 Demand rate variations of schedules meeting nominal separation constraints

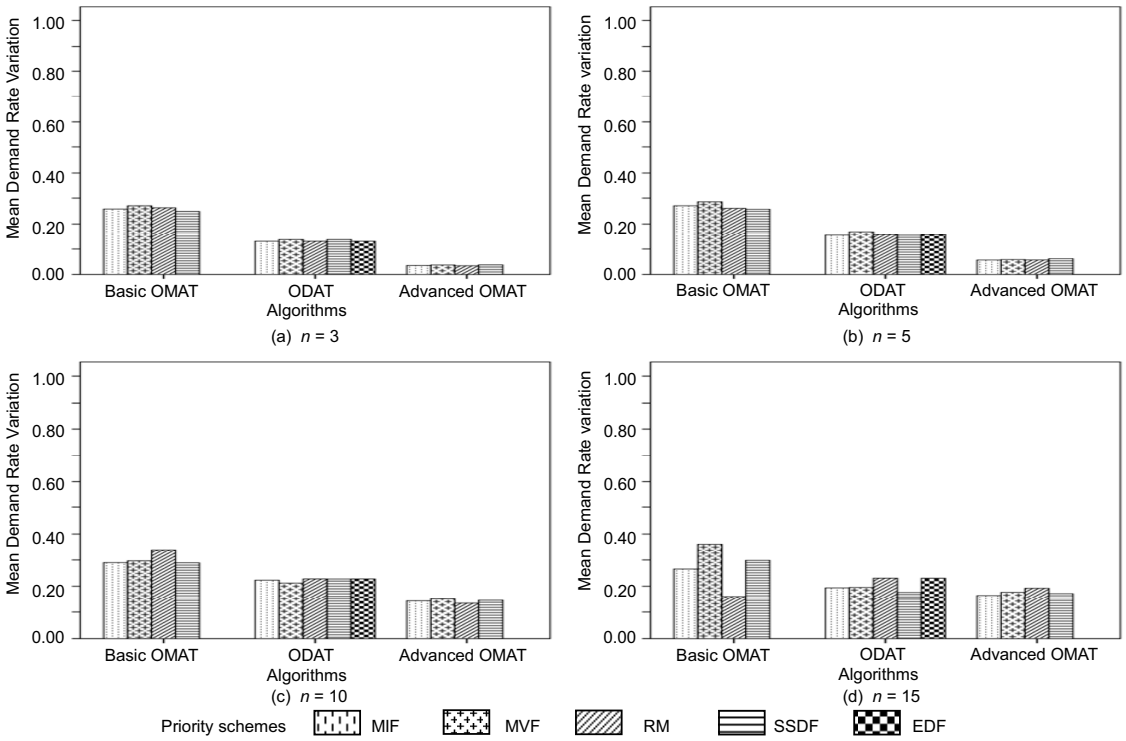

Figure 22 Demand rate variations of schedules meeting absolute separation constraints

6 Summary

This paper focuses on OMAT and ODAT families of scheduling algorithms for multiple interacting medications. OMAT algorithms generate a full schedule for each of the user's medication in turn while ODAT algorithms schedule one dose at a time without prior knowledge, or with limited knowledge, about future doses. A variety of priority assignments are applied on them and refinements are applied as well. Through several simulation experiments, we evaluated the algorithms from two perspectives: success rate and schedule quality. Schedule quality is measured in terms of two criteria, adherence to medication directions and user friendless.

Data obtained from simulation show that performance of the algorithms depends on values of four parameters of MSS: number of medications, dose frequency, interference frequency and interference severity. Specifically, the data show that the advanced OMAT with SSDF priority algorithm has the highest success rate among all evaluated algorithms when the scheduler works with the nominal separation ranges of all medications. However, when the number n of medications increases to more than 10, none of the algorithms can achieve an acceptable success

rate without allowing the scheduler to use the more relaxed absolute separation ranges. When number n of medications is more than 5, the advance OMAT algorithm with MVF priority scheme has the highest success rate among all evaluated algorithms when the scheduler works with absolute separation ranges. One reason is that MVF priority scheme keeps the fraction of time when resources of interfering medications are occupied small. Another reason is that the advanced OMAT algorithm keeps separations between doses large, i.e., keeping the number of doses scheduled small.

In our future works, we will experiment with advanced ODAT algorithm which schedule medications one dose at a time as late as possible. Besides, we take additional constraints into account, including the maximum separations between doses of interfering medications which is also often defined by medication directions. Thus far, we have ignored user preferences. The user may prefer to take medication as infrequently as possible, or take few medications as any dose time, or avoid taking medication in specified time intervals and so on. A smart dispenser typically allows the user to provide these additional scheduling constraints and try to meet them in best effort bases.

Finally, the medication scheduling problem can be expanded to schedule medications of multiple users. Nurses can be offered the efficient route to deliver medications in hospital or nursing home.

Acknowledgement

This work was partially supported by the Taiwan Academia Sinica thematic project SISARL (http://sisarl.org). The authors wish to thanks C. S. Shih, H. C. Yeh and C. Y. Yu for their inputs and suggestions.

References

[1] Veacez, P. J., "An individual based framework for a study on medical error," *International Journal for Quality in Health Care*, Vol. 18, No. 4, May 2006.

[2] "Preventing medication errors," Report Brief, Institute of Medicine of the National Academies, http://www.iom.edu/Object.File/Master/35/943/medication%20errors%20new.pdf

[3] Lisby, M., L. P. Nielsen and J. Mainz, "Errors in the medication process: frequency, type, and potential clinical consequences," *International Journal for Quality in Health Care*, Vol. 17,

No. 1, 2005.

[4] Law, A. V., M. D. Ray, K. K. Knapp, and J. K. Balesh, "Unmet needs in medication use process: perceptions of physician, pharmacists, and patients," *Journal of the American Pharmaceutical Association,* Vol. 43, No 3, 2003.

[5] Wertheimer, A. I, and T. M. Santella, "Medication compliance research," *Jr. of App. Res. in Clinical and Experimental Therapeutics*, 2003.

[6] "Pharmacy – nursing shared vision for safe medication use in hospitals: executive session summary," *Am. Journal of Health – Syst Pharma,* Vol. 60, 2003.

[7] J. W. S. Liu, C. S. Shih, P. H. Tsai, H. C. Yeh, P. C. Hsiu, C. Y. Yu, and W. H. Chang, "Point-of-Care Support for Error-Free Medication Process," in *Proceedings of High Confidence Medical Devices, Software, and Systems (HCMDSS) and Medical Device Plug-and-Play (MD PnP) Interoperability,* June 2007.

[8] http://www.dynamic-living.com/automated_medication_dispenser.htmand http://www.epill.com/

[9] P. C. Hsiu, H. C. Yeh, P. H. Tsai, C. S. Shih, D. H. Burkhardt, T. W. Kuo, J. W. S. Liu and T. Y. Huang, "A General Model for Medication Scheduling," Institute of Information Sciences, Academia Sinica, Taiwan, Technical Report No. TR-IIS-05-008, October 2005.

[10] 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 IEEE Real-Time Systems Symposium*, 2006.

[11] J. W. S. Liu, *Real-Time Systems*, Chapter 1-3, Prentice Hall, 2000.

[12] H. C. Yeh, P. C. Hsiu, C. S. Shih, P. H. Tsai and J. W. S. Liu, "APAMAT: A Prescription Algebra for Medication Authoring Tool," *Proceedings of IEEE International Conference on Systems, Man and Cybernetics,* October 2006.

[13] PDRHealth: http://www.pdrhealth.com/drugs/

[14] P. H. Tsai and J. W. S. Liu, "Consistency and Feasibility of Flexible Demand-Supply Constraints," Submitted to *Journal of Scheduling*, 2007.