Journal of the Society of Korea Industrial and Systems Engineering
Vol. 34, No. 3, pp.16－23, September 2011.

불확실한 도로 네트워크에서의 응급차량 배차

최재영* · 김흥규**†

*동부CNI 연구소
**단국대학교 경영학부

Emergency Vehicle Dispatching on Unreliable Road Networks

Jae Young Choi* · Heung-Kyu Kim**†

*R&D Center, Dongbu CNI
**School of Business, Dankook University

이 논문에서는 지진이나 해일과 같은 자연재해가 발생했을 때 생존자의 수를 최대화하기 위한 응급차량의 배차 문제에 대하여 살펴본다. 이 경우에는 도로 네트워크상에서 최단거리에 있으리라 예상되는 환자부터 실어 나르는 스케줄링 규칙을 많이 이용한다. 이 스케줄링 규칙을 SEPT(Shortest Expected Processing Time)라고 한다. 이 논문에서는 SEPT보다 효율적이라 생각되는 새로운 스케줄링 규칙을 제안한다. 이 스케줄링 규칙은 처리시간과 납기일이 확정적인 경우에 생존자의 수를 최대화시켜주는 스케줄링을 제공하는 Moore의 알고리즘을 처리시간과 납기일이 확률적인 경우로 확장한 것이다. 다음으로 이 스케줄링 규칙을 지진이 많이 발생하는 미국 캘리포니아 주 로스앤젤레스의 한 지역을 대상으로 몇 가지 생존확률 하에서 생존자의 수를 기준으로 SEPT와 비교한다. 그 결과 모든 생존확률 하에서 제안된 스케줄링 규칙이 SEPT보다 평균적으로 생존자의 수를 더 늘려주는 스케줄링을 제공한다. 그리고 많은 경우 제안된 스케줄링 규칙은 최적 스케줄링을 제공한다.

Keywords : Emergency Vehicle Dispatching, Stochastic Scheduling, Number of Late Jobs

1. Instroduction

When natural disasters such as an earthquake or a tsunami strikes, there will be collapses of buildings, broken bridges, fires, and more importantly, many injuries or casualties. The degree of injury varies depending on the emergency situation. Some injuries require immediate medical treatment and others need only minor treatment.

In normal situations, emergency vehicle dispatching rule would be First Come First Served (FCFS). In massive disasters, however, FCFS is not useful because there will be many severe injuries in a very short time period. For this reason, shortest expected processing time (SEPT) rule, in which emergency vehicle is dispatched to the expected nearest injury first, is usually adopted in practice.

In this case, however, the notion of survivability needs to be considered since it incorporates the response time, the time it takes to give the first treatment to an injury, with the severity of the injury. The survivability may well be described as the probability that an injury will survive if an emergency vehicle is dispatched right away. That is, the survivability is the probability that the response time is less than the unknown time of death.

In the trauma literature, the term 'golden hour' is ubiq-

논문접수일 : 2011년 06월 06일 논문수정일 : 2011년 06월 29일 게재확정일 : 2011년 08월 02일
† 교신저자 heungkyu@dankook.ac.kr

uitous, as noted in Lerner and Moscati [2] and in Tallon et al. [6]. The concept is that trauma patients have better outcomes if they are provided with definitive care within 60 minutes after the occurrence of their injuries. It is told that the mortality rate triples for every 30 minute increase from time of injury to definitive care. This can be considered as an example which shows the importance of the response time.

Therefore, in emergency vehicle dispatching, survivabilities as well as response times to the emergency scene must be considered simultaneously. The approach to emergency vehicle dispatching in this paper is borrowed from stochastic machine scheduling literature.

Each injury constitutes a job and its unknown time until death unless treated properly is regarded as a stochastic due date. The sum of the travel time to and from an injury can be expressed as a stochastic processing time. Injuries waiting for treatments are a set of jobs. Finally, every injury is considered to be equally important, which implies, in a scheduling sense, that every job has the same weight.

In the above setting, maximizing the number of survivals is equal to minimizing the number of late jobs. Once the emergency vehicle dispatching decision for an injury is made, the decision is never interrupted until it finishes. Thus each job is non-preemptive.

In the machine scheduling literature, there has not been an efficient method for minimizing the expected number of late jobs when processing times and due dates are distinct. Therefore, in this paper, an efficient method that provides approximated (or near-optimal) solutions is proposed.

The remainder of this paper is organized as follows: Section 2 introduces SEPT rule and proposes a heuristic which is similar to that in Moore [3]. Section 3 develops a framework in which the two scheduling rules are compared and evaluated and Section 4 shows major findings about the two scheduling rules. Final remarks are addressed in Section 5.

2. Scheduling Rules

There are many instances in which the penalty for a late job remains the same no matter how late it is. For example, any delay in the completion of all tasks required for preparation of a new product development would cause the product launch to be aborted, regardless of the length of the delay. As another example, considered in this paper, if an emergency vehicle brings an injury to a hospital after the injury dies, there is no point in dispatching the emergency vehicle no matter how late it is.

In this section, a scheduling rule for dispatching emergency vehicle(s), SEPT, frequently utilized in practice when there are many injuries in a very little time window, is explained in detail after an introduction of some terminologies used in machine scheduling literature. Then a new scheduling rule, devised and proposed in this paper, is fully explained.

Terminology Assume that there are n jobs to be processed through one machine. For each job i, $i=1,2,\cdots,n$, define the following quantities :

p_i = Processing time for job i
d_i = Due date for job i
W_i = Waiting time for job i
C_i = Completion time for job i
L_i = Lateness of job i

The processing time p_i and the due date d_i are characteristics that describe job i. The waiting time W_i for job i is the amount of time that the job must wait before its processing can begin. It is also the sum of the processing times for all the preceding jobs. The completion time C_i for job i is simply the waiting time plus the job processing time.

The lateness L_i of job i is defined as $C_i - d_i$, and thus lateness can be either a positive or a negative quantity. A job is called late when the lateness is positive.

SEPT (Shortest Expected Processing Time)

SPT (Shortest Processing Time) sequences jobs in increasing order of their processing times. The job with the shortest processing time is processed first, the job with the next shortest processing time is processed second, and so on.

In Nahmias [4], it is shown that SPT minimizes the mean completion time when the processing time and the due date are deterministic. It is also shown that the following measures are equivalent : 1) Mean completion time, 2) Mean waiting time, and 3) Mean lateness. Therefore, with the two facts taken together, SPT minimizes mean completion time, mean waiting time, and mean lateness for single machine scheduling.

In addition, in case of parallel processing on identical machines, it has been shown that SPT is an effective rule for minimizing mean completion time. Notice, however,

that SPT does not always minimize the number of late jobs.

In practice it is likely that the exact processing time of one or more jobs may not be predictable. Therefore, it is interesting to know whether or not there are some results concerning the optimal scheduling rules when processing times are uncertain. It is assumed that processing times are independent of one another.

In case of processing on a single machine, the results are quite similar to those discussed for the deterministic case. Assume that the job processing times, P_1, P_2, $\cdots$, P_n, are random variables with known distribution functions. The goal is to minimize the expected completion time; that is,

$$Minimize\, E\left(\frac{1}{n}\sum_{i=1}^{n} C_i\right)$$

Rothkopf [6] has shown that the optimal solution is to schedule the jobs so that job i precedes job $i+1$ if $E(P_i) < E(P_{i+1})$. Notice that this rule, called SEPT (Shortest Expected Processing Time) is simply to process the jobs in increasing order of expected processing times. That is, it is essentially the same as SPT. However, this does not imply that SEPT always minimize the expected number of late jobs.

Somewhat more interesting is when scheduling jobs with random processing times on multiple machines. An assumption that is usually made for the multiple machine problem is that the distribution of job times is exponential. This assumption is needed because the exponential distribution is the only distribution which has the memoryless property. The memoryless property implies that the probability that a job is completed in the next instant of time is independent of the length of time already elapsed in processing the job.

Consider the following situation: n jobs are to be processed through two identical parallel machines. Each job needs to be processed only once on either machine. Assume that at time $t=0$ machine 1 is occupied with a prior job, job 0, and the remaining processing time of job 0 is t_0, which could be deterministic or random. The remaining jobs are processed as follows : let [1], [2], $\cdots$, [n] be a permutation of the n jobs. Job [1] is scheduled on the vacant machine. Job [2] follows either job 0 on machine 1 or job [1] on machine 2, depending on which is completed first. Each successive job is then scheduled on the next available machine.

If the objective is to minimize the expected completion time on two machines which is simply the expected waiting time plus the expected job processing time, then SEPT minimizes the expected completion time on two machines. However, notice that SEPT does not have anything to minimize the expected number of late jobs. For details, see Nahmias [4].

Suppose there are m identical machines, Then the idea behind SEPT for two machine scheduling can be further applied to multiple machine scheduling.

From the discussions so far, it can be easily seen that SEPT does not always guarantee the minimum expected number of late jobs when both processing times and due dates are random.

Algorithm in Moore [3] A polynomial time algorithm for minimizing the number of late jobs, when all the processing times and due dates are deterministic and distinct, is developed in Moore [3]. The problem is to order n jobs, 1, 2, $\cdots$, n, with known processing times, p_1, p_2, $\cdots$, p_n, and due dates, d_1, d_2, $\cdots$, d_n, through a single machine in such a way to minimize the number of late jobs.

It is assumed that each and every job can be finished by its' due date if the processing of the job were to start immediately, that is, $p_i \leq d_i$, $i=1, 2, \cdots, n$. Moore's algorithm consists of a decision rule for dividing the set of jobs, J^c, into two subsets, J and J^d, which is as follows. For details, see Moore [3].

Step 1.
Set $J=\varnothing$, $J^d=\varnothing$, and $J^c=\{1, 2, \cdots, n\}$

Step 2.
Let j^* denote the job with the earliest due date in J^c
Set $J=J\cup\{j^*\}$ and $J^c=J^c-\{j^*\}$
Go to Step 3

Step 3.
If job j^* becomes early, that is $\sum_{i\in J} p_i \leq d_{j^*}$
Go to Step 4
otherwise (j^* becomes late)
Let k^* be the job with the longest processing time in J.
Set $J=J-\{k^*\}$ and $J^d=J^d\cup\{k^*\}$

Step 4.
If $J^c=\varnothing$, STOP
otherwise, go to Step 2.

The algorithm in Moore [3] iteratively includes a job to the set of early jobs J in increasing order of due dates, and if a job becomes late after its inclusion to the set of early jobs, the algorithm discards a job with the longest processing time which has already been scheduled, and includes it to the set of late jobs J^d. The algorithm produces an optimal schedule if the jobs in the set of early jobs J are scheduled according to their due dates, and are followed by the jobs in the set of late jobs J^d in any order.

The complexity of the algorithm in Moore [3] is the same as that of a simple sorting algorithm, which is $O(n\log n)$. Therefore, the algorithm in Moore [3] can be considered as a polynomial time algorithm for minimizing the number of late jobs.

Suppose there are n jobs to be scheduled where each job i has a processing time X_i, an arbitrary independent random variable, that is, any random variable which has non-negative values, and a due date D_i, an arbitrary independent random variable, that is, any random variable which has non-negative values as well. The objective is to minimize the expected number of late jobs. Even if due dates are exponentially distributed, there is no known solution method to get the minimum expected number of late jobs except for total enumeration when processing times and due dates are arbitrary random variables. Therefore, we propose a heuristic that is based on the algorithm in Moore [3] as follows.

Step 1.
Set $J=\varnothing$, $J^d=\varnothing$, and $J^c=\{1, 2, \cdots, n\}$

Step 2.
Let j^* denote the job with the earliest expected due date in J^c
Set $J=J\cup\{j^*\}$ and $J^c=J^c-\{j^*\}$
Go to Step 3

Step 3.
If job j^* becomes early with at least a predetermined chance threshold α, that is
$\Pi_{i\in J}P(X_i \le D_{j^*})\ge \alpha$
Go to Step 4
otherwise (j^* becomes late)
Let k^* be the job such that $\min_{i\in J}P(X_i \le D_{j^*})$.
Set $J=J-\{k^*\}$ and $J^d=J^d\cup\{k^*\}$

Step 4.
If $J^c=\varnothing$, STOP
otherwise, go to Step 2.

The above heuristic iteratively includes a job to the set of early jobs J in increasing order of expected due date, and if a job becomes late in probabilistic sense after its inclusion to the set of early jobs, the heuristic discards a job with the least probability of being early which has already been scheduled, and includes it to the set of late jobs J^d. The heuristic is expected to produce a good schedule if the jobs in the set of early jobs J are scheduled in increasing order of due dates, and are followed by the jobs in the set of late jobs J^d in any order.

Suppose there are m identical machines, Then the heuristic for single machine scheduling can be further extended to multiple machine scheduling as follows.

Step 1.
Set $J=\varnothing$, $J^d=\varnothing$, $J^c=\{1, 2, \cdots, n\}$, and
$J_k=\varnothing$, $S_k=1$, for $k=1, 2, \cdots, m$.

Step 2.
Let j^* denote the job with the earliest expected due date in J^c
Set $J=J\cup\{j^*\}$ and $J^c=J^c-\{j^*\}$
Go to Step 3

Step 3.
Find a machine t such that
$S_t=\max_{k=1, 2, \cdots, m} S_k\left(=\Pi_{i\in J_k}P(X_i \le D_{j^*})\right)$

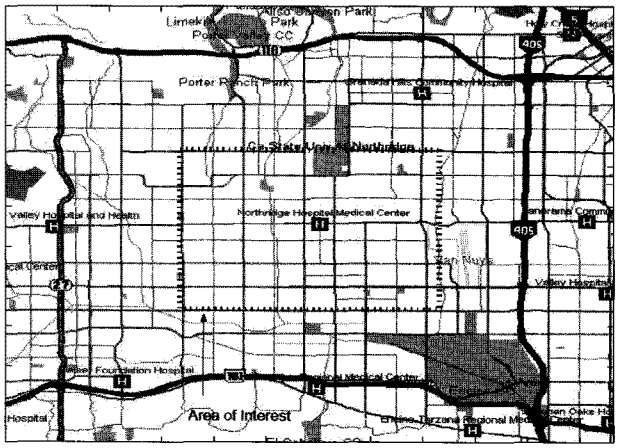
Step 4.
If $S_tP(X_{j^*} \le D_{j^*})<\alpha$, find a job k^* such that $\min_{i\in J}P(X_i < D_{j^*})$
If $k^*=j^*$, then go to Step 5.
Otherwise, replace k^* with j^*.
If $S_tP(X_{j^*} \le D_{j^*})\ge \alpha$, $J_t=J_t\cup\{j^*\}$.

Step 5.
If $J^c=\varnothing$, STOP
otherwise, go to Step 2.

The key idea behind the above heuristic is that it maintains a predetermined chance threshold for each machine. It keeps adding a new job to the machine with a maximum probability of being early. If the product of probabilities reduces below the predetermined chance threshold, it removes the job with the minimum probability among all the machines, and adds the new job to the machine from which a job is removed.

3. Data Preparation

In this paper, emergency vehicle dispatching is considered with geometric data of Northridge in northern Los Angeles, California. This area is as shown in <Figure 1>.



<Figure 1> Area of Interest

Northridge Hospital Medical Center is located in the middle of the area. We assume that an ambulance or emergency vehicle starts at the hospital, arrives at the scene of emergency, and brings an injury back to the hospital.

There are some bridges in the area. Each bridge has a probability of being broken when an earthquake strikes. Those probabilities can be obtained by HAZUS which is a decision tool for estimating natural hazards loss developed by FEMA (Federal Emergency Management Agency). Taking those probabilities into account, the travel time from one point in the map to another is adjusted. The detailed algorithm to compute the adjusted travel times is based on Monte Carlo simulation. For a given pair of source and destination points, a shortest path algorithm finds a shortest route taking into account traffic status such as speed limits, traffic amount, and so on as shown in Frank [1]. In Frank [1], the shortest path algorithm to obtain the adjusted travel time is extensively exploited. For each randomly generated scenario, the shortest travel time is computed by the shortest path algorithm and the adjusted travel time is the average of those travel times. Therefore, the average of those travel times, which we call HAZUS travel times, can be used in place of the expected travel times for the given pair of source and destination points. For details, see Frank [1].

The HAZUS travel times are computed by a decision support system developed at Center for Multi-Information Fusion, University at Buffalo. For detailed algorithm for HAZUS travel times, visit http://www.infofusion.buffalo.edu/.

The maximum travel time within this area from the hospital is about 11 minutes. If we have 10 injuries in this area, it will take up to 220 minutes for a single emergency vehicle to pick up and deliver all the injuries to the hospital. We set the number of injuries to 10. For 10 injuries, we have 10 sets of uniformly selected injury locations within the area of interest, which are as shown in <Table 1>.

<Table 1> HAZUS Travel Times for Experiments

Location Set	HAZUS Travel Times				
1	1.4498	2.9590	4.9445	6.0987	6.4353
	6.9359	8.7660	9.3874	9.8894	10.2596
2	1.5499	4.0817	4.0885	5.3309	5.4341
	5.5663	8.1010	8.1526	8.2780	10.2191
3	1.6584	1.8993	2.6136	3.8947	5.9684
	6.0515	7.7102	7.8955	8.3116	9.6485
4	2.0443	2.1051	2.7093	3.3747	3.9148
	4.0905	8.1346	8.1978	8.3393	10.4754
5	2.1305	2.5529	2.7900	3.3180	3.5214
	6.2692	7.0646	7.5005	8.1335	9.2597
6	2.2785	2.6463	2.8588	4.7313	6.5902
	6.6218	6.9756	7.0858	7.5742	9.5021
7	2.2770	2.8835	3.5064	4.1387	4.6464
	6.7188	8.0028	8.2778	9.5913	9.6983
8	1.5313	3.3388	4.0807	5.0219	5.8318
	6.2398	6.4337	6.6888	7.2210	10.2400
9	1.5911	2.1721	3.2331	3.8639	5.0611
	7.6834	8.2070	9.5756	10.0054	10.0971
10	1.8323	2.9580	3.0694	5.6123	7.2351
	7.7444	7.8417	7.9642	9.2288	10.9305

It is assumed that the due date of injury i follows an exponential distribution with rate δ_i. The due date represents the severity of each injury. The survivability, that is, the probability that the processing time is less than or equal to the due date, is randomly chosen. Let X_i, D_i and s_i be the HAZUS travel time, the due date and the survivability of injury i, respectively. Then the rate δ_i can be easily obtained by the following relationship.

$$s_i = P(2X_i \le D_i) = e^{-2\delta_i X_i}$$

It would be interesting to know how many injuries can survive according to their distribution of survivabilities. Five categories of survivabilities are prepared, which are characterized by their minimum survivabilities which are

0.5, 0.6, 0.7, 0.8 and 0.9. For example, if minimum survivability is 0.5, the survivability for each injury is randomly generated by a Uniform Distribution, $U(0.5, 1)$. For each combination of location set and minimum survivability, there are 30 random samples generated. Therefore, there are 1,500($= 10 \times 5 \times 30$) random samples in all.

4. Computational Experiments

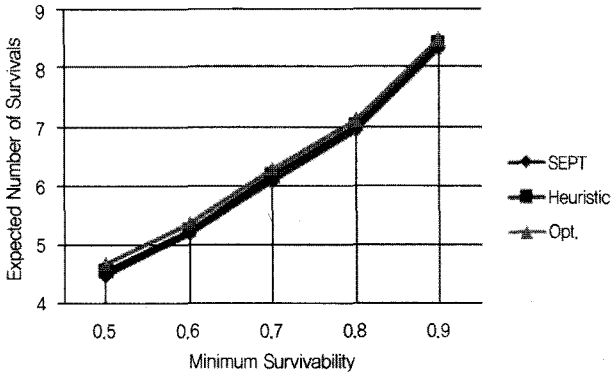
As mentioned in Section 1, the goals of this paper are to compare and evaluate the two rules for emergency vehicle dispatching, SEPT and the heuristic based on the algorithm in Moore [3], in terms of the expected number of survivals. It can be easily seen that SEPT rule simply ignores the severity of injuries. It takes only HAZUS travel times into account and sequences injuries in increasing order of HAZUS travel times. For the heuristic, by varying thresholds from 0.001 to 0.999 in increments of 0.001, the early injuries are selected. The late injuries are sequenced after the early injuries in increasing order of HAZUS travel times. This way of emergency vehicle dispatching using the heuristic chooses the best schedule that maximizes the expected number of survivals among the 1000 chance thresholds.

Processing on a Single Machine : For each level of minimum survivability, these two rules are tested on 10 different injury location sets. As shown in <Table 2>, the heuristic outperforms SEPT. Each element in <Table 2> shows the total average of the expected number of survivals in 10 different injury location sets. In <Table 2>, the optimal solutions are obtained by total enumerations. The number of all the permutations of 10 jobs is 10! = 3,628,800.

<Table 2> SEPT vs. Heuristic

Min. Survivability	SEPT	Heuristic	Opt.
0.5	4.4654	4.5407	4.6776
0.6	5.1904	5.2466	5.3656
0.7	6.0872	6.1726	6.2821
0.8	6.9383	7.0291	7.1178
0.9	8.3461	8.4231	8.4753

Surprisingly, SEPT performs well when minimum survivability is low, which can be also found in <Figure 2>. This gives an important insight : when the severity of injuries is not known, it is good enough to schedule injuries in increasing order of HAZUS travel times, taking the expected nearest injury first.



<Figure 2> Comparison of Performances

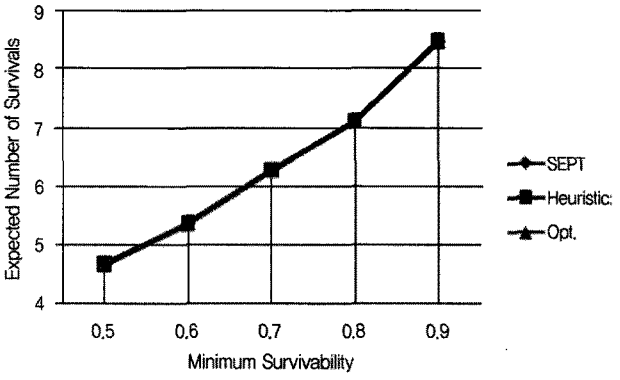
Adjacent Pairwise Interchange (API) is also applied to these two rules. API repeatedly looks for adjacent pairs of jobs that would improve the given schedule if they are interchanged. API stops when there are no more adjacent pairs of jobs that improve the schedule.

The heuristic outperforms SEPT for each level of minimum survivability. As shown in <Table 3>, the heuristic with API generates near optimal solutions. Each element in <Table 3> shows the total average of the expected number of survivals in 10 different injury location sets. As the minimum survivability increases beyond 0.8, the heuristic with API generates optimal solutions. Even when the minimum survivability is 0.5, the performance of the heuristic with API is very close (> 99%) to the optimal solutions.

<Table 3> Performances of the Heuristic with API

Min. Survivability	Heuristic with API	Opt.	Performance Ratio
0.5	4.6557	4.6776	0.9953
0.6	5.3577	5.3656	0.9985
0.7	6.2815	6.2821	0.9999
0.8	7.1178	7.1178	1.0000
0.9	8.4753	8.4753	1.0000

Both of the two rules exhibit very similar performances to each other as shown in <Figure 3>. The performance of SEPT is quite promising when API is employed. Although the heuristic is a little better than SEPT, SEPT with API performs fairly well and it will generate good solutions when the severity of injuries is not known.



<Figure 3> Comparison of Performances with API

Parallel Processing on Identical Machines : The two rules are compared using the same data sets described in Section 3.

SEPT for multiple machines schedules the jobs in increasing order of processing times as in the case of a single machine. It repeatedly assigns jobs to the machine with the shortest expected completion time.

The optimal schedule is obtained by total enumeration of all the possible permutations of jobs. For each permutation, it assigns jobs one by one in the schedule to the machine with the shortest expected completion time. Then, it computes the expected number of late jobs for each permutation. Among all the permutations of jobs, the optimal schedule has the minimum expected number of late jobs.

As shown in <Table 4>, <Table 5>, the heuristic outperforms SEPT and is very close to the optimal schedule (up to 99.95%) on average. The CPU times for both SEPT and the heuristic are all negligible.

<Table 4> Number of Survivals : 3 Vehicles

Min. Survivability	SEPT	Heuristic	Opt.
0.5	6.4209	6.4746	6.5650
0.6	7.1013	7.1697	7.2405
0.7	7.8050	7.8881	7.9378
0.8	8.3935	8.4575	8.4908
0.9	9.2033	9.2374	9.2555

<Table 5> Number of Survivals : 5 Vehicles

Min. Survivability	SEPT	Heuristic	Opt.
0.5	7.0157	7.0536	7.1057
0.6	7.6250	7.6733	7.7078
0.7	8.2368	8.2853	8.3099
0.8	8.7267	8.7619	8.7778
0.9	9.3808	9.3979	9.4061

5. Final Remarks

This paper uses geometric data in an area of interest to compare and evaluate two emergency vehicle dispatching rules. The area of interest is located at Northridge in northern Los Angeles, California. Any geometric point in the area can be reached within at most 11 minutes from the hospital which is located in the middle of the area. It is assumed that a single emergency vehicle delivers 10 injuries one by one.

Two efficient emergency vehicle dispatching rules are examined in this paper. These rules are based on stochastic scheduling problems in which the expected number of late jobs is minimized.

First, SEPT, widely used in practice, schedules jobs in increasing order of processing times.

Next, a heuristic that approximates the exact optimal solutions is proposed. The chance threshold is varied from 0.001 to 0.999 for each run. At each run, the heuristic is used to schedule the jobs. Finally, a sequence with the maximum expected number of survivals is chosen as an approximate to optimal solutions.

Comparing those two rules, SEPT and the heuristic, the heuristic outperforms SEPT. It is noteworthy, however, that SEPT is very promising when the severity of injuries is not known.

When API is applied to the two rules, the performances are quite similar to each other. However, still, the heuristic provides better performance. Since API requires information about the severity of injuries, it is not useful when there is no information on the severity of injuries.

The two rules for multiple vehicles are compared as well. With three and five vehicles, the heuristic is very close to the optimal solutions with negligible amount of CPU time.

These findings imply that we need to consider the heuristic for emergency vehicle dispatching rather than SEPT to save more injuries until an exact method for optimal solutions is found. Since it is expected for us to experience more natural disasters in the near future, the need for using better emergency vehicle dispatching rule will be increasing.

References

[1] Frank, W.; Hazardous Material Vehicle Routing Utilizing a Decision Support System and Temporal Road Attributes. Ph.D thesis, University at Buffalo, State University

of New York, 2001.

[2] Lerner, E. B. and Moscati, R. M.; The "Golden Hour," Scientific Fact or Medical Urban Legend?, *Academic Emergency Medicine*, 8(7) : 758-760, 2001.

[3] Moore, M. J.; An N Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs, *Management Science*, 15(1) : 102-109, 1968.

[4] Nahmias, S.; Production and Operations Analysis, Irwin, Homewood, IL, 1993.

[5] Rothkopf, M. S.; Scheduling with Random Service Times, *Management Science*, 12 : 707-713, 1966.

[6] Tallon, J. M., Lerner, E. B., and Moscati, R. M.; The "Golden Hour," *Paradigm, Academic Emergency Medicine*, 9(7) : 760-760, 2002.