Transport Planning with Service-Level Constraints: Complexity and Algorithms

Hoong Chuin LAU and **Kien Ming NG** and **Xiaotao WU**
The Logistics Institute-Asia Pacific
National University of Singapore, 119260

Abstract

In this paper, we study a problem arising in military transport planning. A military organization operates a large fleet of vehicles in a depot to serve the requests of various operational units. Each request has a fixed start and end time, and is served by a prescribed number of vehicles. We address the following two problems: (1) how many vehicles are at least needed to meet a given service level of requests; and (2) suppose we allow each request to shift its start time by a constant duration, can all the requests be met?

Introduction

We consider a real-world transport logistics planning problem arising in a military organization. It is easy to see that this problem is also applicable in a logistics agency which services multiple clients, where service level constraints play an important role. The organization owns a large fleet of vehicles centralized in a depot to service requests from multiple operational units. A request has a start and end time, a weight and requires a number of vehicles. For simplicity, we assume that all vehicles are identical. The problem is to minimize the number of vehicles needed to meet a given service level constraint, or to maximize the number of served requests using a given number of vehicles. In the case where not all requests can be satisfied, a related interesting question is whether, by allowing some requests to be *shifted time-wise*, all requests can be satisfied with a certain fixed number of vehicles. The latter question is particularly important in the military context in achieving 100% operational readiness. Military commanders are interested to know whether, within a reasonable delay to their operations, they are able to guarantee full supply of resources. Without loss of generality, we assume that requests are to be shifted to a later time point.

Fig. 1 shows an example of a request of size (quantity) 2 vehicles, with start time $s = 1$ and end time $t = 3$. Fig. 2 gives an example of an input instance of 4 requests with same weights. Fig. 3 shows two schedules using 2 vehicles, where the first schedule is able to satisfy only 3 requests with no shifts, while the second schedule satisfies all 4 requests with an allowable shift of 1 time unit.

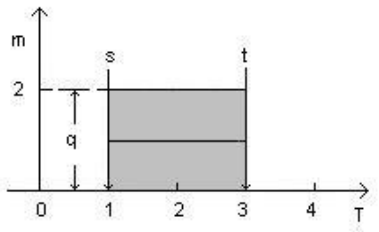

Figure 1: Example of a request: $q = 2, s = 1, t = 3$

	quantity	start time	end time
A	1	0	1
B	1	1	2
C	1	1	3
D	2	2	4

Figure 2: Input instance of 4 requests: A,B,C,D

Literature Review

In this section, we review the literature for related problems. Our problem is a generalization of what is commonly known as the problem of scheduling jobs with fixed start and end times. In graph theory, this problem

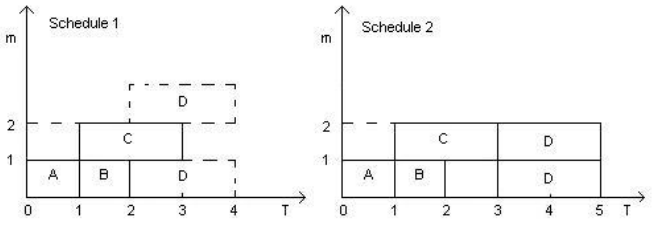

Figure 3: Two schedules. Schedule 1 satisfies requests A,B,C while Schedule 2 satisfies requests A,B,C,D with 1 unit shift of D

is a generalization of the Dilworth's problem of finding a minimum chain decomposition of a given graph, which can be solved via a min-cost flow formulation. Examples of work in this area include the work of Arkin and Silverberg (AS87) and more recently, the online algorithms presented by Woeginger (W94). Most of the literature emphasizes on jobs with single-resource requirement, because the problem exhibits very nice combinatorial structure (such as the property that the underlying matrix is totally unimodular). In this paper, we consider the multi-resource problem, and treat the single-resource problem as a special case.

In standard machine scheduling terminology, jobs are to be started anytime within job-specific release times and completed before their deadlines. To our knowledge however, little work has been done in the special case of allowing *constant* shifts of job start time, which, to some extent, seems to be an easier problem comparatively. In this paper, we show that this problem is NP-complete, even for shift of a single unit of time on jobs with equal release time and duration. This result is tight in the sense that the problem can be solved in polynomial time when jobs have single-resource requirement, even when they have arbitrary release times and due-dates (S83). Another example of work done in allowing shifts is due to Gertsbakh and Stern(GS78), who proposed an integer programming approach for the single-resource problem. There is a large collection of unreported related machine scheduling problems with release times and deadlines (see for example Brucker's collection at (BK)), but suffices to say that most of these papers deal with different objective functions such as makespan and job-specific release times, which are not the concerns of this paper.

Military Transportation Planning

For convenience, we say requests are *unweighted* when all the weights of the requests are equal, and *weighted* otherwise. A *unit* (or *single-vehicle*) request refers to one whose quantity of vehicles required is exactly 1. In this paper, unless otherwise specified, we consider the general problem of weighted, multi-vehicle requests.

We formally define the Military Transportation Planning (MTP) problem as the following optimization problem: Given a set of requests $R = \{1, \ldots, n\}$ indexed by i over a planning period T and an integer k, find the minimum of vehicles needed to satisfy at least k requests. This problem can be easily extended to the weighted case where the goal is to satisfy requests whose total weight is at least k. The dual optimization problem dMTP is also interesting: given an integer m, find the maximum weight (or number) of requests that can be satisfied using not more than m vehicles.

Mathematical Programming Formulation

Table 1 shows the notations that will be used in the formulation as well as subsequent discussion of the MTP problem. We denote duration of request i as $l_i = t_i - s_i$, and the number of the vehicles required by all the requests as $h = \sum_{i=1}^{n} q_i$.

Type	Symbol	Description
Input Variable	T	number of time periods in the planning horizon
Input Variable	n	total number of requests over the planning horizon
Input Variable	k	number (weight) of requests to be fulfilled
Input Variable	q_i	quantity of vehicles required by request i
Input	s_i	start time of request i
Input	t_i	end time of request i
Input	w_i	weight of fulfilling request i
Decision Variable	$x_{i\ell}$	number of vehicles fulfilling request i $\{0 \leq i \leq n\}$ and then request ℓ $\{1 \leq \ell \leq n+1,\ \ell \neq i\}$
Decision Variable	y_i	$y_i = 1$ if request i is fulfilled; 0 otherwise

Table 1: Mathematical Notations

Note that we need to introduce two additional "dummy" requests, i.e., request 0 and request $n+1$, to reflect the respective start and end of any feasible vehicle assignment to the physical requests. Thus, for any vehicle j, a feasible assignment to the requests would be of the form $0 \to j_1 \to j_2 \to \cdots \to j_p \to n+1$, where $p,\ j_1,\ j_2,\ \ldots,\ j_p \in \{1,\ 2,\ ,\ldots,\ n\}$, or $0 \to n+1$ indicating that the vehicle is not used at all. The mathematical programming formulation is then:

$$min \sum_{i=1}^{n} x_{0i} \tag{1}$$

such that

$$\sum_{\substack{\ell=1 \\ \ell \neq i}}^{n+1} x_{i\ell} \geq q_i y_i \quad (i = 1, 2, \ldots, n) \tag{2}$$

$$\sum_{\substack{\ell=0 \\ \ell \neq i}}^{n} x_{\ell i} - \sum_{\substack{r=1 \\ r \neq i}}^{n+1} x_{ir} = 0 \quad (i = 1, 2, \ldots, n) \tag{3}$$

$$\sum_{i=1}^{n} w_i y_i \geq k \tag{4}$$

$$(s_\ell - t_i) x_{i\ell} \geq 0 \quad (i = 1, 2, \ldots, n; \ell = 1, 2, \ldots, n,\ i \neq \ell) \tag{5}$$

$$x_{i\ell} \in \mathbb{Z}_{\geq 0} \quad (i = 0, 1, \ldots, n; \ell = 1, 2, \ldots, n+1,\ i \neq \ell) \tag{6}$$

$$y_i \in \{0, 1\} \quad (i = 1, 2, \ldots, n). \tag{7}$$

Equation (1) reflects the aim of minimizing the total number of vehicles used. Equation (2) ensures that a request is fulfilled only if the required number of vehicles are assigned to fulfil that request. Equation (3) is a conservation constraint ensuring that the total number

of vehicles assigned for a particular request remains the same both before and after fulfilling a request. Equation (4) ensures that the weighted total number of requests satisfied meets the required level. Equation (5) is a constraint forcing variables $x_{i\ell}$ corresponding to an invalid sequence of fulfilling requests to assume a value of zero. Equations (6) and (7) are integrality constraints.

The number of integer decision variables required is easily verified to be $O(n^2)$ for this formulation.

Complexity and Algorithms

In this section, we present an NP-completeness result and show that certain special cases of MTP can be solved in polynomial time.

NP-completeness of MTP The MTP decision problem is defined by: given n requests $R\{1,\ldots,n\}$, each request with start time s_i, end time t_i, weight w_i and quantity q_i, integer m and k, is there an m-vehicle schedule satisfying requests whose total weight are at least k?

It is easy to prove that the MTP problem is NP-complete via a reduction from KNAPSACK which is defined by: Given a finite set U of items, each item $u \in U$ having a size $s(u) \in \mathbb{Z}^+$ and a value $v(u) \in \mathbb{Z}^+$, integers $B \in \mathbb{Z}^+$ and $K \in \mathbb{Z}^+$, is there a subset $U' \subseteq U$ such that

$$\sum_{u\in U'} s(u) \leq B \text{ and } \sum_{u\in U'} v(u) \geq K$$

KNAPSACK is known to be NP-complete. We reduce a KNAPSACK instance to a MTP instance as follows. For each item i, create a request i with $s_i = 0, t_i = 1$, and set q_i and w_i as the size and value of item i.

Relationship between MTP and dMTP Given a set of requests R, we can define parameterized MTP and dMTP problem instances as follows:

$$MTP(k) = min\{f(x)|g(x) \geq k\}. \quad (8)$$

$$dMTP(m) = max\{g(x)|f(x) \leq m\}. \quad (9)$$

where $f(x)$ and $g(x)$ are functions on a schedule x counting the number of vehicles used and the number (weight) of requests satisfied respectively. Let x^* and y^* respectively denote an optimal solution for problems (8) and (9).

The following theorem relates problem instances (8) and (9).

Theorem 1 *$f(x^*) = m$ if and only if:*

- *(A) $g(y^*) \geq k$; and*
- *(B) for all feasible schedule y of dMTP($m-1$), $g(y) < k$.*

Proof Only if part: If m is the optimal value for MTP(k), then by definition there is no y with $f(y) < m$ such that $g(y) \geq k$. Hence condition (B) is implied, and by setting $y^*=x^*$, condition (A) holds.

If part: First note that the maximum value of dMTP(m) is monotonically non-decreasing in m since more requests can be satisfied with more vehicles. By condition (B), there is no y with $f(y) < m$ such that $g(y) \geq k$. Hence, by condition (A), m is the least value for which there exists a solution y with $g(y) \geq k$, implying that $f(x^*) = m$ if x^* is an optimal solution. ∎

This suggests that if we can solve the problem dMTP(m), then simply by iterating from 1 to m and checking if the number of satisfied requests has reached k, we can get the optimal value of problem MTP(k). However, note that in the multi-vehicle case, since the value q_i of each request i is unbounded, this method suffers high complexity if some requests require large number of vehicles. Whereas, for single-vehicle case where $m = O(n)$, this method allows us to solve MTP by solving at most n instances of dMTP.

Conversely in a similar fashion, if we can solve the problem MTP(k), then simply by iterating from n down to 1, we can use the algorithm to solve dMTP(m). This allows us to solve dMTP by solving at most n instances of MTP, even for the multi-vehicle case.

Special Case 1: $k = n$ This problem is to decide the minimum number of vehicles so that ALL requests can be fulfilled. We present a polynomial time algorithm to output the schedule if one exists.

We consider the unit-request MTP problem first. Let each request be an interval on the real line whose endpoints are specified by the start and end times of the request. Left (Right) endpoint indicates the start (end) time. Each instance of unit-request MTP can be modeled as an *interval graph*. Thus the problem can be formulated in terms of *interval graph coloring problem*.

Definition Interval Graph Coloring Problem. Consider a set of intervals on a line. If we represent these intervals by an undirected graph $G(V, E)$ such that two vertices in V are connected by an edge in E iff the two corresponding intervals overlap, then G is an *interval graph*. The Interval Graph Coloring Problem (IGC) is to decide the minimum of colors needed to color all vertices of a given interval graph.

The problem can be solved optimally in polynomial time, by known IGC algorithms (such as (G72)). Furthermore, since all requests need to be satisfied eventually, each multi-vehicle request can be converted into multiple unit requests. Hence, MTP can be optimally solved by simply applying any known IGC algorithm. The time complexity is $O(h^2)$ using interval graph coloring algorithms, since to construct the interval graph itself, we need time $O(h^2)$.

Further algorithmic improvement is possible. In fact, the MTP problem can be solved by applying a plane-sweep algorithm directly and the resulting time complexity is $O(n \log n + h)$ using $O(h)$ space: Generate and sort the endpoints of all intervals in non-decreasing order. For each endpoint picked from the sorted linked list from left to right, if it is a left endpoint of request

i, allocate q_i free vehicles for it and declare the q_i vehicles busy; if it is a right endpoint of i, similarly release all the q_i vehicles back to the list of free vehicles. For sorting the list, we need $O(n \log n)$ time; for each endpoint associated with request i, we need $O(q_i)$ time to allocate/deallocate vehicles by querying two bit vectors of size at most h. The total running time is hence $O(n \log n + h)$ using $O(h)$ space.

Special Case 2: $m = 1$ When there is only 1 vehicle (and therefore all requests are unit requests), dMTP is reduced to the Activity Selection Problem, which can be solved in $O(n)$ time via a greedy algorithm if endpoints are presorted : At each step, select a feasible activity (request) with earliest finish time with endpoints presorted such that it is compatible with already selected activities (requests). Even for the weighted case, dMTP can be solved by formulating it as a Shortest Path problem and applying Dijkstra's algorithm(D59). With the corresponding graph constructed, which has $|E|$ edges and $|V|$ nodes, the complexity of Dijkstra's algorithm is $O(|E|+|V| \log |V|)$(FT84). In our problem, with the interval graph constructed, $|V|$ is $O(n)$ and $|E|$ is $O(n^2)$, and hence the time complexity is $O(n^2)$.

Special Case 3: Unit-requests For the unit-request problem, we note that the minimum number of vehicles required is bounded from above by the number of requests n. Here, two algorithmic results can be achieved.

1. In (AS87), Arkin and Silverberg proposed a $O(n^2 \log n)$ time algorithm to solve the problem of scheduling n jobs, with fixed start and end times. Their problem is to maximize the value of scheduled jobs on m identical machines for any given m. Their approach is to model it as a min-cost flow problem. The above problem is actually the unit-request dMTP problem. Hence, by applying the same approach, the dMTP problem on a given value of m can be solved by min-cost flows in $O(n^2 \log n)$ time (PS82). Note that the algorithm also implicitly solves all related dMTP instances greater than m, implying that when we choose $m = 1$, by Theorem 1, MTP can be solved with one iteration in $O(n^2 \log n)$ time.

2. When $m = 2$, the above situation can be improved. An $O(n^2)$ algorithm was proposed by Hsiao *et al.* (HTC92) based on dynamic programming to solve the *maximum weight 2-independent set problem* on weighted interval graphs. Their approach can be generalized to $O(n^m)$, for finding an m-independent set on weighted interval graphs. As the dMTP problem on m is equivalent to finding a maximum (weight) m-independent set, it means that an $O(n^2)$ algorithm is found for $m = 2$, and $O(n^m)$ for arbitrary m. This means, by Theorem 1, that MTP can be solved in $\sum_{m=1}^{m^*} O(n^m) = O(n^{m^*})$ time. Thus, MTP instances with a minimum value of $m^* \leq 2$ can be solved in $O(n^2)$ time.

MTP with Shifting

We define the MTP with Shifting (MTP-S) problem as: given n requests over a planning period T, and an integer C, find a schedule that satisfies ALL requests by allowing each request's start time to be shifted later by a duration not exceeding C time units.

Mathematical Programming Formulation

By introducing more variables than that used in MTP, it is also possible to model MTP-S in a similar way. Since we can shift the start (and therefore the end) time by C, this is equivalent to each request having a release time r_i and deadline d_i where $d_i = r_i + l_i + C$, for $1 \leq i \leq n$. See Table 2 for the required changes in the variable definitions.

Type	Symbol	Description
Input Variable	C	maximum allowable shift for request start time
Input	r_i	release time of request i
Derived	d_i	deadline of request i
Decision	s_i	actual start time of request i

Table 2: Additional Notations for MTP-S formulation

It is then necessary to add the following constraints on the start time of each request:

$$r_i \leq s_i \leq r_i + C \quad (i = 1, 2, \ldots, n) \tag{10}$$

$$s_i \in \mathbb{Z}_{\geq 0} \quad (i = 1, 2, \ldots, n). \tag{11}$$

Since we would like to have all requests being satisfied, equation (4) can be omitted and (2) changed to

$$\sum_{\substack{\ell=1 \\ \ell \neq i}}^{n+1} x_{i\ell} = q_i \quad (i = 1, 2, \ldots, n) \tag{12}$$

Also, equation (5) would be modified to:

$$(s_\ell - s_i - l_i) x_{i\ell} \geq 0 \; (i = 1, 2, \ldots, n; \ell = 1, 2, \ldots, n, \; i \neq \ell). \tag{13}$$

As equation (13) is nonlinear in the decision variables, we simplify it further so that we will only need to deal with linear constraints, at the expense of introducing more integer intermediate variables. In particular, define $z_{i\ell}$, for $1 \leq i, \; \ell \leq n$ and $i \neq \ell$ to be binary intermediate variables satisfying the following constraints:

$$x_{i\ell} \leq M z_{i\ell} \; (i = 1, 2, \ldots, n; \ell = 1, 2, \ldots, n, \; i \neq \ell) \tag{14}$$

$$s_\ell - s_i - l_i \geq M(z_{i\ell} - 1) \; (i = 1, 2, \ldots, n; \ell = 1, 2, \ldots, n, i \neq \ell), \tag{15}$$

$$z_{i\ell} \in \{0, 1\} \; (i = 1, 2, \ldots, n; \ell = 1, 2, \ldots, n, i \neq \ell), \tag{16}$$

where $M = \max\{\max_i\{q_i\}, T\} + 1$. It is easy to verify that equations (14) and (15) could be used in place of equation (13), thereby removing the nonlinearities in the formulation of MTP-S.

In summary, the mathematical programming formulation of MTP-S is described by the minimization of

equation (1) subject to the constraints (3), (6), (7), (10), (11), (12), (14), (15), and (16). The number of integer decision variables required is again $O(n^2)$ for this formulation. This formulation is similar in spirit to that of the vehicle routing problem with time windows (VRPTW) commonly seen in literature, such as (FJM97). Thus, it may be possible to adapt existing methods for handling the VRPTW to solve MTP-S. One such approach is to apply Lagrangian relaxation as in (DDSS99). Following that approach, we find that if we relax constraints (14) and (15), we have

$$min \sum_{i=1}^{n} x_{0i} + \sum_{i=1}^{n} \sum_{\substack{\ell=1 \\ \ell \neq i}}^{n} \lambda_{i\ell} x_{i\ell} + M \sum_{i=1}^{n} \sum_{\substack{\ell=1 \\ \ell \neq i}}^{n} (\mu_{i\ell} - \lambda_{i\ell}) z_{i\ell} \tag{17}$$

$$+ \sum_{i=1}^{n} \sum_{\substack{\ell=1 \\ \ell \neq i}}^{n} (\mu_{i\ell} - \mu_{\ell i}) s_i + \sum_{i=1}^{n} \sum_{\substack{\ell=1 \\ \ell \neq i}}^{n} \mu_{i\ell} (l_i - M)$$

such that

$$\sum_{\substack{\ell=1 \\ \ell \neq i}}^{n+1} x_{i\ell} = q_i \quad (i = 1, 2, \ldots, n) \tag{18}$$

$$\sum_{\substack{\ell=0 \\ \ell \neq i}}^{n} x_{\ell i} - \sum_{\substack{r=1 \\ r \neq i}}^{n+1} x_{ir} = 0 \quad (i = 1, 2, \ldots, n) \tag{19}$$

$$r_i \leq s_i \leq r_i + C \quad (i = 1, 2, \ldots, n) \tag{20}$$

$$x_{i\ell} \in \mathbb{Z}_{\geq 0} \quad (i = 0, 1, \ldots, n; \ell = 1, 2, \ldots, n+1, \ i \neq \ell) \tag{21}$$

$$y_i \in \{0, 1\} \quad (i = 1, 2, \ldots, n). \tag{22}$$

$$z_{i\ell} \in \{0, 1\} \quad (i = 1, 2, \ldots, n; \ell = 1, 2, \ldots, n, i \neq \ell), \tag{23}$$

where $\lambda_{i\ell}$ and $\mu_{i\ell}$ are the Lagrangian multipliers of the respective constraints and could only take on nonnegative values. As the Lagrangian relaxation problem is separable in the variables $x_{i\ell}$, $z_{i\ell}$ and s_i, we can easily deduce that it has the Integrality Property by observing that the subproblem involving $x_{i\ell}$ is a network flow problem, while

$$z_{i\ell} = \begin{cases} 1, & \text{if } \mu_{i\ell} < \lambda_{i\ell}, \\ 0, & \text{otherwise,} \end{cases}$$

$$s_i = \begin{cases} r_i, & \text{if } \sum\limits_{\substack{\ell=1 \\ \ell \neq i}}^{n} (\mu_{i\ell} - \mu_{\ell i}) > 0, \\ r_i + C, & \text{otherwise.} \end{cases}$$

This implies that the solution obtained by this Lagrangian relaxation is simply that of the LP relaxation of the MTP-S, and so would not necessarily be a good lower bound of the optimal objective of MTP-S. It is also possible to formulate other Lagrangian relaxation problems and this will be a subject of further study.

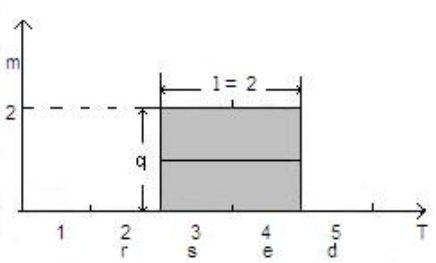

Figure 4: A request example

Complexity

We now prove that the unweighted MTP-S decision problem is NP-complete, even when $C = 1$ and all release times and durations are equal. Recall MTP-S problem: Given n requests $R = \{1, \ldots, n\}$ indexed by i, each request with release time r_i, duration l_i, integer m and constant C, is there an m-vehicle schedule over planning period T for R, that meets all the time constraints, i.e. $r_i \leq s_i \leq r_i + C$ and $s_i + l_i \leq d_i$?

Proof It is known that the 2-Partition problem is NP-complete: Given a finite set A and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$. Is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$? We can restrict MTP-S to the 2-Partition problem by considering the instances in which $h = \sum_{i=1}^{n} q_i = 2n$; $C = 1$; $r_i = 0$ and $l_i = 1 \forall i$; and $m = n$. Since each request can only be served within interval [0,1] or [1,2], MTP-S is equivalent to the problem: given a finite set of requests R and a "size" $q_i \in \mathbb{Z}^+$ for each $i \in R$. Is there a subset $R' \subseteq R$ such that $\sum_{i \in R'} q_i = \sum_{i \in R - R'} q_i$? ∎

Genetic Algorithm for MTP-S Problem

In this section, we propose a genetic algorithm (GA) as a heuristic to solve the NP-hard MTP-S problem. GAs are local search algorithms designed to mimic the process of natural selection and evolution in nature. We assume that the readers are familiar with the terminology for GA.

Before describing our proposed GA, we introduce the notion of *request histograms*.

Request Historgrams

We assume that the total planning period T is composed of t discrete unit periods. The actual start and end periods of each request i are denoted s_i and e_i respectively, where $e_i = s_i + l_i - 1$.

For instance,a request of vehicle quantity $q = 2$, with release period $r = 2$, deadline $d = 5$, length $l = 2$ and allowable shift $C = 2$. Suppose the actual start period is $s = 3$ and actual end period $e = s+q-1 = 3+2-1 = 4$. (See Fig. 4)

We define a *Request Histogram* as one whose x-axis is the time period of interest and whose y-axis is the number of vehicles needed. Corresponding to each period ℓ

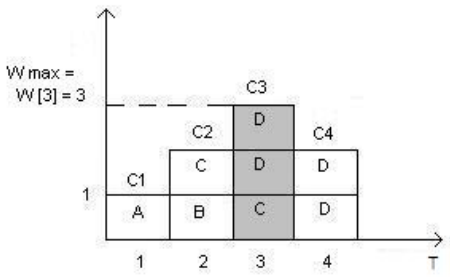

Figure 5: Request Histogram and $CRH : W = (1, 2, 3, 2), W_{max} = W[3] = 3$

is column C_ℓ on the *Request Histogram*. We denote the height of column C_ℓ as h_ℓ and $H_{max} = \max_{1 \le \ell \le t} h_\ell$.

Since only the height of any column has real physical meaning, we need not care about the position of requests of that column. Let $X[t]$ be the *Configuration of Request Histogram* (CRH), where $W[\ell] = h_\ell$ for $1 \le \ell \le t$. Clearly, $W_{max} = \max_{1 \le \ell \le t} W[\ell] = H_{max}$. Fig. 5 shows the *Request Histograms* whose *Configuration* is: $W = (1, 2, 3, 2)$ and $W_{max} = 3$.

Claim 1 *The vehicles assigned to any request need not be consecutive in the schedule.*

Lemma 1 *For each CRH with actual start period s_i of each request i, we can construct a feasible schedule such that the number of vehicles used Sch_{max} is equal to W_{max}.*

Proof Note that if the actual start period s_i of each request i is fixed over t time periods:

1. we can generate a $CRH : W[t]$ such that $W[\ell] = \sum_{i \in P} q_i$, where $P = \{i | s_i \le \ell \le e_i, 1 \le i \le n, 1 \le \ell \le t\}$.
2. we can generate a feasible schedule using an algorithm in $O(n \log(n))$ time. The number of vehicles used, Sch_{max}, equals to the maximum number of vehicles processed on certain time periods over the whole planning horizon T.

For each $CRH : X[t]$ over t time periods, at least $W[\ell]$ vehicles are needed to fulfull the requests on any period ℓ, where $1 \le \ell \le t$. Thus, at least W_{max} are needed to fulfill all the requests. It is then trivial to see that $W_{max} = Sch_{max}$. This means that for each CRH, given the actual start period s_i of each request i, we can at least construct a feasible schedule such that $Sch_{max} = W_{max}$. ∎

One way to find the optimal CRH is to list all the possible CRHs given any instance of MTP-S. This is equivalent to listing all the different combinations of the actual start period s_i of each i and the search space is exponential (C^n). The optimal CRH* has the property that: $W^*_{max} = \min\{W^k_{max} | 1 \le k \le C^n\}$, where W^k_{max} is the value of W_{max} for CRH$_k$. By Lemma 1, we can generate a optimal schedule for the MTP-S instance.

Niche GA for MTP-S

It is known that the simple genetic algorithm(SGA) is suitable for searching the optimum of unimodal functions in a bounded search space. Both experiments and analysis show that the SGA cannot find the multiple global maxima of a multimodal function(GB89)(M95). This limitation can be overcome by a mechanism that creates and maintains several subpopulations within the search space in suah a way that each maximum of the multimodal function can attract one of them. These mechanisms are referred to as "niching methods"(M95).

Our MTP-S problem is modeled as a multimodal function which has only only one optimal solutions. In fact, those CRH with the same W_{max} can be viewed as the same solutions, for the number of vehicles needed is the same with these CRH solutions. Although we do not need to list all the optimal CRHs, since the search space is large, we still need a method to control the search efficiency. We adapt the niching idea here because it is useful to maintain population diversity and permit genetic algorithms to explore more search space so as to identify multiple peaks, whether optimal or otherwise.

There are several niching techniques in the history. The fitness sharing method was proposed by Goldberg and Richardson in 1987 (GB87)and the crowding method by DeJong in 1975. The clearing idea used in this paper derives from the clearing procedure stated by Petrowski in 1996. The basic idea is: instead of evenly sharing the available resources among the individuals of a subpopulation, the clearing procedure supplies these resources only to the best individuals of each subpopulation. In fact, we propose a variation of clearing method which not only concerns the current population but the previous population as well. Thus, we need not worry about the crossover operation destroying the individuals located on the peaks found with a high probability.

Coding The chromosomes are of length n and each gene has the value range in $[0, 1, ...C]$, where C refers to the allowable time shift in the requests. This implies that the total solution space size is C^n.

It is easy to implement using the above coding with the CRH technique. Originally, we just need to construct a CRH of the MTP problem with no time shift using an array$[t]$ with array$[\ell] = W[\ell]$, for $1 \le \ell \le t$. Given any chromosome, the value of the ith gene indicates the number of delay days of the ith request, where $1 \le i \le n$. We just need to adjust the CRH model and we can then get the X_{max} of any given chromosome easily.

Fitness The objective value $f(X)$ for a particular chromosome X is the number of vehicles needed to fulfill all the n requests when that chromosome is applied.

Unlike typical genetic algorithms, we desire chromosomes with *low* objective values, since these correspond to smaller number of vehicles needed. We can simply define the fitness value $F(X)$ using:

$$F(X) = \begin{cases} C_{max} - f(X) & \text{if } f(X) < C_{max} \text{ ;} \\ 0 & \text{otherwise.} \end{cases} \quad (24)$$

Mutation Here, the uniform mutation operation is used. Suppose the chromosome is $X = x_1 x_2 \dots x_k \dots x_\ell$ and x_k is the mutation point with range within [0,C]. After the uniform mutation operation, the new chromosome is $X = x_1 x_2 \dots x'_k \dots x_\ell$ where

$$x'_k = r \times C, \quad (25)$$

and r is a random number from [0,1].

Algorithm The following is a pseudo-code for our Niche GA. The function $g(a, b) = 1$ if $a = b$ and 0 otherwise.

Algorithm 1 ***NicheGA***
begin
1. ***Initialize***
a. $t \leftarrow 1$
b. Generate initial population $P(t)$ with M individuals
2. ***Preparation***
a. Compute the fitness F_i of every individual of P(t) (i = 1,2,...,M)
c. Sort them in decreasing order
b. Record N first individuals. ($N < M$)
3. ***GA Operations***
a. $P(t) \leftarrow P'(t)$ *(Selection operation)*
b. $P(t) \leftarrow P''(t)$ *(Crossover operation)*
c. $P(t) \leftarrow P'''(t)$ *(Mutation operation)*
4. ***Niche clearing operation***
a. Combine N individuals from ***2*** *and M individuals from* ***3*** *into a new population with n = N+M individuals, sorted in decreasing order; Compute Hamming distance of each pair X_i and X_j, i.e. for $1 \le i < j \le n$:*

$$||X_i - X_j|| = \sqrt{\sum_{k=1}^{n} g(X_{ik}, X_{jk})} \quad (26)$$

b. ***if*** *$||X_i - X_j|| < L$), i.e. X_i and X_j are in same niche, assign a penalty to X_k of smaller fitness ($k = i, j$)*
c. Recompute the fitness of all n individuals and sort them in decreasing order
d. Record the first M individuals
*5.****Judgement***
if *Meet stop conditions*
Output result; Jump to ***end***
else
$t \leftarrow t + 1$
Replace P(t) with M new individuals from ***4***
Jump to ***2***
end

Experimental Results

We experiment on two classes of data: random data and practical data obtained from a military organization.

Random Data Generation and Results

Here, we generate data set with known optimal solutions to verify the effectiveness of our algorithm. The procedures of generating one dataset cluster is listed below:

1. Generate a rectangular histogram, (that is the optimal CRH for a dataset cluster)
2. Partition the rectangle into small rectangles as requests (with random durations and quantities)
3. Shift each generated request with k units forward, where $k \in [0, C]$ to generate the start and end time
4. Change C and go to step 3 in order to generate next dataset for this dataset cluster.

In our case, we set each dataset cluster to include 5 datasets, for example, $S1$ dataset cluster consists of $S1\_C1$ to $S1\_C5$ whose C value ranges from 1 to 5, where the rectangle is the optimal solution. Those datasets within the same data cluster share at least one identical optimal solution, which is the big rectangle.

We experimented on 3 datasets types (small, medium and large), each type consisting of 4 dataset clusters. For the small datasets with request size < 20, our GA obtained optimal solutions for all cases. For the medium datasests with request size < 100, our GA also obtained very good solutions whose objective value is 1 or 2 from the optimal solutions. In the interest of space, these results will not be reported.

Table 3 shows the results for large datasests with request size < 500. We compare our NicheGA results with the standard GA and the optimal solutions. Each algorithm is run 5 times and the average values are reported.

The GA parameters used as follows: Population Size: 50, Maxgeneration 1000, $P_c = 0.5$, $P_m = 0.005$; Penalty: C_{max}; for NicheGA, L(Hamming distance): $para1$; N: Population Size / $para2$. (C_{max} is a predefined large number related to the max number of vehicles needed; $para1$ is a parameter related to request size n and shift delay C ; $para2$ is 2.)

Practical Data Generation and Results

We obtained a base case from a military organization and generated practical datasets around that case by carefully considering the distributions of data values on the number of time slots, the number of vehicles, the start and end times of a request, the request quantity, as well as the total number of jobs spanning over the vehicle resources.

Table 4 gives the results of small dataset (size=240,T=365 days). The maximal delay is 5 days. From the table we can see that our NicheGA performs well on all datasets. Here, the optimal solutions were obtained by a branch-and-bound exact

Data Set	Size	Optimal	Niche GA	Standard GA
L1_C1	122	40.0	43.0	43.2
L1_C2	122	40.0	43.0	44.0
L1_C3	122	40.0	43.4	44.6
L1_C4	122	40.0	43.6	45.3
L1_C5	122	40.0	44.0	45.8
L2_C1	245	40.0	43.0	44.0
L2_C2	245	40.0	44.2	45.0
L2_C3	245	40.0	45.0	46.2
L2_C4	245	40.0	44.8	47.2
L2_C5	245	40.0	45.2	47.0
L3_C1	348	40.0	44.0	44.7
L3_C2	348	40.0	44.6	45.9
L3_C3	348	40.0	45.0	47.0
L3_C4	348	40.0	45.5	47.3
L3_C5	348	40.0	46.2	48.3
L4_C1	451	40.0	44.0	45.0
L4_C2	451	40.0	45.0	46.0
L4_C3	451	40.0	45.8	46.9
L4_C4	451	40.0	46.0	48.0
L4_C5	451	40.0	46.0	48.0

Table 3: Large Random Datasets

approach on the mathematical programming model described in Section 4.1.

Delay	Optimal	Niche GA	% of Optimality
C=1	20	20	100%
C=2	18	18	100%
C=3	19	19	100%
C=4	16	16	100 %
C=5	16	16	100 %

Table 4: Small Practical Datasets

Finally, we show the results of testing on a real-life large dataset obtained from a military organization containing 2732 requests over a 2-year (T=730) planning horizon. There are two types of vehicles resources. The results are summarized in Table 5. Here, the parameters refer to the maximum shift value C while GA results show the breakdown of the number of vehicles required for each respective type. The exact result is obtained by solving the mathematical programming model described in Section 4.1 with an early termination of the solution process in order to shorten the time required to obtain a feasible solution.

C	Exact result	GA result	Lower bound
0	39+81 = 120	39+ 81 = 120	-
1	35+73 = 108	35+73 = 108	106
2	32+71 = 103	31+70 = 101	97

Table 5: Real Dataset

References

E. M. Arkin , E. B. Silverberg, "Scheduling jobs with fixed start and end times", Disc. Appl. Math, 18(1), 1-8, 1987

P. Brucker and S. Knust, http://www.mathematik.uni-osnabrueck.de/research/OR/class/

J. Desrosiers, M. Sauvé and F. Soumis, "Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows", Management Sc., 34(8), 1005-1022, 1988

E. W. Dijkstra, "A note on two problems in connexion with graphs", Numer. Math., 1, 269-271, 1959

M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms", Proc. 25th IEEE Symp. on FOCS, 1984

M.L. Fisher, K.O. Jörnsten, O.B.G. Masden, "Vehicle Routing with Time Windows: Two Optimization Algorithms", Oper. Res., 45(3), 488-492, 1997

I. Gertsbakh and H. I. Stern, "Minimal Resources for Fixed and Variable Job Schedules", Oper. Res., 26(1), 68-85, 1978

F. Gavril, "Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph", SIAM J. Comput. 1(2), 180-187, 1972

D.E.Goldberg, J.Richardson, " Genetic Algorithms with Sharing of Multimodal Function Optimization", Proc. 2nd Int'l Conf. Genetic Algo., pp42-49, 1987

D.E.Goldberg, "Genetic algorithms in search, optimization and machine learning", Addison Wesley,1989

J. Y. Hsiao, C. Y. Tang, R. S. Chang, "An Efficient Algorithm for Finding a Maximum Weight 2-Independent Set on Interval Graphs", Info. Proc. Lett., 43, 229-235,1992

KA De Jong, "An analysis of the behavior of a class of genetic adaptive systems", PhD dissertation, University of Michigan, 1975

S.W.Mahfoud,"Niching Methods for Genetic Algorithms", PhD dissertation, University of Illinois Urbana-Champaign, 1995

C. H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", Readings, Prentice-Hall, 1982

A. Petrowski, "A Clearing Procedure as a Niching Method for Genetic Algorithms", IEEE 3rd Int'l Conf. Evol. Comput. (ICEC), Nagoya,1996

B. Simons, "Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines", SIAM J., Computing, 12(2), 294-299, 1983

G. J. Woeginger, "On-line scheduling of jobs with fixed start and end times", Theo. Comp. Sci., 130, 5-16, 1994