Transport Logistics Planning with Service-Level Constraints

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 logistics 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? A Niche genetic algorithm, together with a hybridized variant, are applied to the problem.

Introduction

The world is increasingly service-oriented and hence service constraints play an important role in business. In the military world, service constraints play an ever-increasing role in today's heightened security threats. In this paper, we consider a real-world transport logistics planning problem arising in a military organization. This problem is also prevalent 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.

In this paper, we study the complexity and algorithms for solving 2 major problems in this arena. We formulate the problems as integer programming models which render an exact solution approach, such as branch-and-bound (B&B), to solve them to optimality. However, such an exact solution approach may not be practical when tackling large problem instances as the solution process may take an exponential amount of time. It is also possible to apply genetic algorithms (GA) on the problems and solutions could then be obtained for large problem instances. But GAs may encounter the situation of being trapped in local optimality. To overcome the shortcoming of both approaches, we propose in this paper a hybrid algorithm that hybridize GA and B&B.

In the interest of space, the proofs of theorems and lemmas have been omitted in this version.

Literature Review

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 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 & Silverberg 1987) and more recently, the online algorithms presented in (Woeginger 1994). 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 started and completed within job-specific release times and 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 (Simons 1983). Another example of work done in allowing shifts is due to (Gertsbakh & Stern 1978), 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 collection at (Brucker & Knust)), 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 number 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.

We use a 2-D bar-chart to represent the requests, where the x-axis represents time, and the y-axis vehicles. Each request is represented as a horizontal bar from its start time to end time with height equals to the number of vehicles required.

Mathematical Programming Formulation

Suppose there is a total of n requests over a planning horizon of T time periods, and we would like to fulfill k requests, where $0 \leq k \leq n$. We let q_i, s_i, t_i, and w_i be the quantity of vehicles required, the start time, the end time and the weight of request i respectively, where $0 \leq i \leq n$. We define our decision variables to be $x_{i\ell}$ that represents the number of vehicles fulfilling request i and then request ℓ, where $0 \leq i, \ell \leq n, i \neq \ell$; and y_i that takes a value of 1 if request i is fulfilled, and 0 otherwise, where $0 \leq i \leq n$. We also denote the duration of request i as $l_i = t_i - s_i$, and the number of the vehicles required by all the requests as $h = \sum_{i=1}^{n} q_i$.

Here, we 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:

$$min \sum_{i=1}^{n} x_{0i} \qquad (1)$$

such that

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

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

$$\sum_{i=1}^{n} w_i y_i \geq k \qquad (4)$$

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

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

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

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.

NP-Completeness of MTP

The MTP decision problem is NP-complete via a reduction from the Knapsack problem which is known to be NP-complete.

Relationship between MTP and dMTP

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

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

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

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 problems (8) and (9).

Theorem 1 *$f(x^*) = m$ if and only if: (1) $g(y^*) \geq k$; and (2) for all feasible schedules y of dMTP$(m-1)$, $g(y) < k$.*

This theorem 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 from high complexity issues if some requests require a large number of vehicles. For the single-vehicle case where $m = O(n)$, this method allows us to solve MTP by solving at most n instances of dMTP.

Conversely, 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: 100% service level This problem is to decide the minimum number of vehicles so that ALL requests can be fulfilled. The unit-request MTP problem can be solved by coloring of an interval graph, whose nodes represent the requests and edges represent overlapping time constraint. Interval graph coloring can be solved optimally in polynomial time, by known algorithms (such as (Gavril 1972)). Furthermore, since all requests need to be satisfied eventually, the multi-request MTP problem can easily be solved by treating each multi-vehicle request as multiple unit requests. The problem can be solved by applying a plane-sweep algorithm and the resulting time complexity is $O(n \log n + h)$ 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 (Dijkstra 1959). With the corresponding graph constructed, which has $|E|$ edges and $|V|$ nodes, the complexity of Dijkstra's algorithm is $O(|E| + |V| \log |V|)$ (Fredman & Tarjan 1984). 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 results can be established.

1. (Arkin & Silverberg 1987) proposed a $O(n^2 \log n)$ time algorithm to solve the problem of scheduling n jobs with fixed start and end times that maximizes 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 (Papadimitriou & Steiglitz 1982). 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 in (Hsiao *et al.* 1992) 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. As dMTP on m is equivalent to finding a maximum (weight) m-independent set, an $O(n^2)$ algorithm is found for $m = 2$, and in general, an $O(n^m)$ algorithm 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 (MTP-S)

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.

The unweighted MTP-S decision problem is NP-complete, even when $C = 1$ and all release times and durations are equal. The proof is a reduction from the 2-Partition problem which is known to be NP-Complete.

Mathematical Programming Formulation

By introducing more variables, we can 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, deadline d_i and an actual start time s_i, where $d_i = r_i + l_i + C$ and l_i is the duration of request for $1 \leq i \leq n$.

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:

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

If we define $z_{i\ell}$, for $1 \leq i, \ell \leq n$ and $i \neq \ell$ to be binary intermediate variables satisfying the following constraints:

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

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

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

where $M = \max\{\max_i\{q_i\}, T\} + 1$ then it is easy to verify that equations (14), (15) and (16) 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 (Fisher *et al.* 1997). Thus, it may be possible to adapt existing methods for handling the VRPTW to solve MTP-S, such as the Lagrangian relaxation approach in (Desrosiers *et al.* 1988).

Genetic Algorithm for MTP-S Problem

In this section, we propose a genetic algorithm (GA) to solve the NP-hard MTP-S problem.

Cumulative Request Histogram Before presenting our proposed GA, we introduce the notion of a *request histogram.* By aggregating the 2-D graph representation of all requests, we derive a *Cumulative Request Histogram*, whose x-axis is the time period and y-axis is the total number of vehicles required by requests spanning each time period. Let h_ℓ denote the number of vehicles required at time period l. Let CRH = $(h_1, \ldots, h_\ell, \ldots, h_T)$ and $CRH_{max} = \max_{\ell=1}^{T} h_\ell$.

Lemma 1 *For each CRH with fixed actual start times (i.e. no shift allowed), we can construct a feasible schedule such that the number of vehicles used m^* is equal to CRH_{max} in polynomial time.*

This lemma is evident by inspecting the plane sweep algorithm presented in Special Case 1 that we need at most CRH_{max} vehicles. The proof of optimality results from the observation that we need at least CRH_{max} vehicles through the pigeon-hole argument. The computational complexity is $O(n \log n + h)$, as presented above.

The optimal schedule CRH^* for MTP-S has the property that $CRH^*_{max} = \min\{CRH^k_{max} | k \in S\}$, where S is the set of all possible CRH resulting from fixing the actual start times of requests, and CRH^k_{max} is the value of CRH_{max} for the k-th CRH. Hence, by Lemma 1, we may generate a optimal schedule by simply enumerating all possible combinations of actual start times. Unfortunately, there is an exponential number (precisely C^n) of such combinations. In this paper, we adopt a Niche genetic algorithm which reduces the search space efficiently.

Niche GA for MTP-S

Both experiments and analysis show that while the simple genetic algorithm is suitable for searching the optimum of unimodal functions in a bounded search space, it cannot find the multiple global maxima of a multimodal function (Goldberg 1989). This limitation can be overcome by a mechanism that creates and maintains several subpopulations within the search space in such a way that each maximum of the multimodal function can attract one of them. These mechanisms are referred to as "niching methods" in (Mahfoud 1999).

Our MTP-S problem is modeled as a multimodal function which has many optimal solutions. In fact, those CRHs with the same CRH_{max} can be viewed as the same solutions, for the number of vehicles required is the same. Although we do not have to list all the optimal CRHs, since the search space is large, we still need a method to control the search efficiency. We adopt 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 literature. The fitness sharing method was proposed by Goldberg and Richardson (Goldberg & Richardson 1987) and the crowding method by DeJong (Jong 1975). The clearing idea used in this paper derives from the clearing procedure stated by Petrowski (Petrowski 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 the 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 For the chromosome $\lambda = \{j_1, \ldots, j_i, \ldots, j_n\}$, where $j_i \in [0, 1, \ldots, C]$, n is the number of requests and C refers to the allowable time shift in the requests. This implies that the total solution space size is C^n.

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

Selection & **Crossover** We use the *Roulette Wheel Selection* as our basic model (Goldberg 1989). Also and *elitist* model is then used for comparison where the best performing chromosome is exempt from mutation. We use *one-point crossover* in Niche GA.

Mutation Here, the *uniform mutation* operation with probability p_m is used.

Suppose the chromosome is $\lambda = \{j_1, j_2, \ldots, j_n\}$ and p is the selected mutation point, $1 \leq p \leq n$. After the uniform mutation operation, the new chromosome is $\lambda = \{j_1, j_2, \ldots, j'_p, \ldots, j_n\}$ where $j'_p = r \times C$ and r is a random number from [0,1].

Hybrid Approach

A pure Niche GA approach may still be inadequate in obtaining optimal or close-to-optimal solutions for certain problem instances. Here, we propose a hybrid algorithm that adapts the Niche GA method to make use of the strengths of an exact method. With the mathematical programming formulation, there are certainly many exact solution approaches that can produce the optimal solution though at the expense of computation time. For simplicity, we assume that our exact solution approach is a B&B method that is always generating feasible solutions with improving objective values if we apply it to the MTP-S problem. We start by running both Niche GA and B&B algorithms on an instance of the MTP-S problem. Each time the B&B algorithm returns a new feasible solution that is better than the current best solution of the Niche GA, we convert it into a corresponding chromosome, say λ^*.

For each gene j_i of λ^*, we calculate the sum of hamming distance with that of all the chromosomes of the current population, and call it $dis(i)$, for $1 \leq i \leq n$. We then change the Niche GA parameters so that the resulting Niche GA algorithm will attempt to generate genes that are similar to that obtained by the B&B method. Thus, we can set the mutation probability $p_m[i]$ of each gene position to be $\frac{dis(i)}{PopSize}$, for $1 \leq i \leq n$, where $Popsize$ is the size of the genetic population pool. But sometimes we do not want the offspring of the current generation to be similar as λ^*, especially at certain time periods where vehicles are densely scheduled. We can simply avoid this by setting a high mutation probability to the corresponding requests, i.e. $p_m[i] = 1$, for $1 \leq i \leq n$. We then proceed with running both the Niche GA and the B&B algorithms, and repeat the process with each new feasible solution returned by the B&B algorithm. The algorithm is terminated if either the B&B algorithm produces an optimal solution, or a prespecified iteration time limit is reached.

The reason for such a hybrid approach is that we are using the exact solution approach as a diversification strategy for the Niche GA algorithm. There has been literature on diversification strategies for GA algorithms (see e.g. (Ting *et al.* 2001)), but most of them are using heuristic or randomized techniques. A strong justification for our proposed approach is that we are in effect using an exact solution approach, which is guaranteed to obtain an optimal solution to the problem, to guide the diversification strategy. This will help prevent the Niche GA algorithm from getting iterates that are trapped in a local optimal solution. We can also interpret the process as the Niche GA algorithm "learning" from the intelligent search procedures used by an exact solution approach.

Experimental Results

We experimented on two classes of data: practical data obtained from a military organization and randomly generated data.

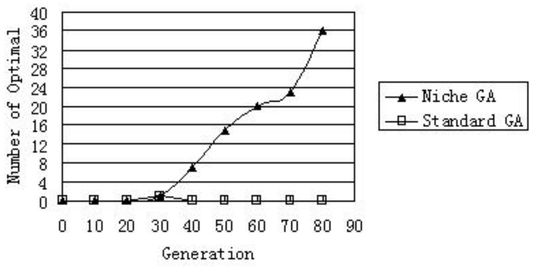

Figure 1: Population spread of Niche GA and Standard GA, with population size of 50

Practical Data Generation and Results

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

We compare the Niche GA and the exact integer programming (IP) model proposed above on a small data set D1 (n=240, T=365 units), with delay C ranging from 1 to 5. Although both Niche GA and IP model obtain the optimal solution, the time taken for the IP model increases exponentially as the delay value increases, compared with linear scaleup for Niche GA.

We also did a comparison of the Niche GA and the Standard GA, using the same genetic operators. Figure 1 shows the number of individuals with optimal solutions as it varies from generation to generation. It can be seen that both Niche GA and Standard GA get the optimal solution in an early generation (both when generation = 30) due to the use of the CRH technique which reduces the solution space. Also, we can see that the Niche GA has a much higher likelihood of obtaining other optimal solutions than the Standard GA.

Random Data Generation and Results

Here, we generate data sets with known optimal solutions to verify the effectiveness of our Niche GA. The procedures of generating one data set cluster are:

1. Generate a rectangular histogram (that is the optimal CRH for a data set 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 the next data set for this data set cluster.

We set each data set cluster to include 5 data sets, whose mean value of C ranges from 1 to 5. It is trivial to see that these 5 data sets within the same data cluster share at least one identical optimal solution, which is the optimal CRH generated.

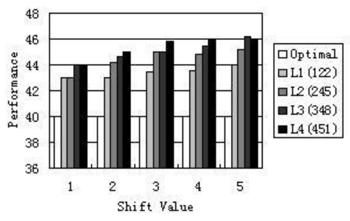

Figure 2: Performance of Niche GA on large data sets

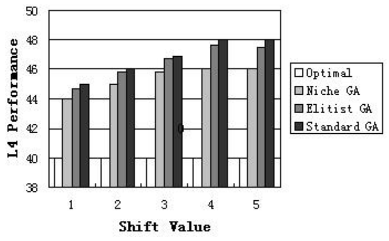

Figure 3: Performance of different GAs on L4

We experimented on 3 types of data sets (small, medium and large), with each type consisting of 4 data set clusters. Figure 2 shows the performance (number of vehicles required) of Niche GA on large data clusters $L1$ ($n = 122$), $L2$ ($n = 245$), $L3$ ($n = 348$), $L4$ ($n = 451$). It can be seen that Niche GA does not perform satisfactorily especially when C is large. We also compared Niche GA with other GAs on data cluster $L4$. The average performance over 5 runs are reported in Figure 3. Compared with standard GA and Elitist GA, the Niche GA performs slightly better. In such cases, the search space is too large compared to the number of near optimal solutions. It is very hard to get the optimal solution because of very few optimal solutions (the worst case may only have one optimal CRH, i.e. with no symmetric cases included).

Given the poor performance of the Niche GA approach, we apply our proposed hybrid algorithm on the test data. In particular, we experimented with a test data set with size 62 requests and a shift of 5 time units, and it has an optimal solution of 20 vehicles. The pure Niche GA method is unable to get a solution that is better than 22 vehicles, while the B&B method takes a large number of iterations to obtain a solution that is less than 23 vehicles. With the hybrid algorithm, a solution of 21 vehicles can be obtained.

Conclusion

In this paper, we studied a class of logistic problems. We have also developed mathematical programming models that allow exact solution approaches to produce intermediate solutions for intelligently influencing the GA's parameters. Future work is to extend this hybridization scheme to other classes of problems.

References

E. M. Arkin and 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", Mgmt. 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", Reading, 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

K. A. 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", Reading, Prentice-Hall, 1982

A. Petrowski, "A Clearing Procedure as a Niching Method for Genetic Algorithms", Proc. 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

C.-K. Ting, S.-T. Li, C. Lee, "TGA: A new integrated approach to evolutionary algorithms", Proc. Congress on Evolutionary Computation, 917-924, 2001

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