FLEET SIZING PROBLEM WITH TIME WINDOWS USING GENETIC ALGORITHM

Nguyen Thi Quyen[1], Nguyen Thi Hien[2*], Hoang Van Hai[3], Nguyen Long[4]

Abstract: *This paper presents solution for fleet sizing problem with time windows (FSPTW) using genetic algorithm (GA). FSPTW involves identifying the optimal number of vehicles in environments with shuttle transportation tasks with known demands and predefined time windows. Those environments are very common place business settings where items are transferred again and again between more than one machines by means of a fleet of vehicles. Usual examples of such environments are manufacturing factories, warehouses and container ports. Two container ports were selected as case studies for this research. Results show that the suggested approach is quiet effective, as it provides solutions that are competitive with the best known in the literature.*

Keywords: Evolutionary computation; Genetic algorithm; Fleet sizing problem.

1. INTRODUCTION

In this paper, we propose a GA to identify the optimal number of vehicles in environments with shuttle transportation tasks (ESTTs). In [1] ESTTs are representative of industrial settings where items are transferred repeatedly between multiple pickup-and-delivery points (PDPs) by a fleet of vehicles and there is a machine to performs the items at each PDP. A certain vehicle will pick up items after they have been processed and transfer to other machine to process if needed. There might be a buffer beside each machine, which a finite area is determined to temporarily store items in a queue. The reason for the buffer is to diminish delayed time. Vehicles can drop off items in the buffer without waiting for the machines to could accessible. Machines also can put the items on the buffer for vehicles to gather after. ESTTs are very common in industrial execution. Regular cases are manufacturing factories, warehouses and container terminals (CTs) [1]. In an ESTT, the fleet sizing problem (FSP) is the crucial problem, of which the purpose is to determine the optimal number of vehicles to carry items. Using too few vehicles can reduce achievement while having too many vehicles is costly and may launch deadlocks. The FSPTW specification requires a minimization of the number of vehicles in the situations where the solution space is definite by narrow time windows. From a real-world point of view, this may be depend on many factors such as uncertainty in travel time of vehicles, the processing time of machines and size of the buffer. In fleet sizing problems (FSP), the objective for the optimal search is the minimum number of vehicles to pick up goods from a source point to a destination point (delivery point). But in the real world, one other objective is still concerned is the time cost for each solution of total number of vehicles. These approaches [1] seek for approximate solutions in a graph representation using evolutionary algorithm. However, using a graph to model the solution of FSP may be increase the arduous. Therefore, we proposed a method to solving FSP by obvious way is possible to consider problem further. The rest of the paper is organized as follows. We first present related work on FSP with time windows constraint in section 2. Section 3, 4 outlines notion of FSP and the method. We

describe the experimental settings in section 5, presenting and discussing the results in section 6. We provide overall conclusions in section 7, and highlight directions for future work.

2. RELATED WORKS

There are some proposals for FSP recently, in [1], the authors developed an evolutionary algorithm able to identify the suitable number of vehicles as single objective in environments with shuttle transportation tasks (ESTTs), in both static and uncertain situations.

In [2], the authors presented an analytical model for estimating vehicle requirement using regression analysis. The experimental results show that the regression model is a promising technique for estimating vehicle requirements. The results obtained from test problems provide comparable performance between regression and simulation approaches. The vehicle requirements predicted by the model differed from the results obtained from the simulation model only slightly.

The paper indicated that regression analysis provides an alternative and promising approach for estimating vehicle requirements for an automated guided vehicle system (AGVs), especially during the initial phase of a system design. In [3] the authors modeled the movement of AGVs as a multiclass CQN accounting for the loaded and empty travel times as well as loading, unloading, and blocking time. They modeled the steady-state behavior of the closed queuing network (CQN) by a linear program whose optimal value was the minimal fleet-size. The fleet size recommended by the LP model was compared with the actual fleet size used in the simulation models, and in only three instances out of 24 instances was the suggestion off by one AGV. The authors in [4] proposed a new model of AGVs arising from modern transportation system, and propose an approximately analytical method to study the minimum vehicle number required when system is stable. A numerical example based on simulation is also provided to illustrate the approximately analytical method. In [5], some models were developed to estimate the empty trip distribution under two popular dispatch rules. The paper indicated that empty trip information can significantly improve the performance of the final AGVs design and that even a simple model of empty trip travel can capture much of the effect. An integer programming model in [3] has been proposed to address the AGVs capacity requirements planning problem. The model minimizes the overall system cost defined as the sum of the annualized cost of operating the AGVs and the cost of transporting parts between workstations considering the required number of part movements between workstations and vehicle capacities. In [6] an integer linear programming model is introduced, which can be efficiently solved by any optimization solver such as CPLEX. A numerical example has been carried out for the model assessment and impact analysis of the confidence parameters and cargo shipment demand.

In [7], the authors developed a fuzzy linear dynamic model of rail car flows (loaded and empty) in a network which is the basis for using the optimal control theory to solve for the fleet size to support a total minimum cost solution. The proposed model includes a specification of uncertain demands as fuzzy random

variables. The paper indicated that the optimal control theory can be employed for determining the size of rail freight car fleet in fuzzy stochastic case, based on establishing balance between the exact cost parameters, the cost of unmet demand, on one hand, and the sum of car ownership and utilization costs, on the other. In [8], the authors proposed a formulation and a solution procedure for optimizing the fleet size and freight car allocation problem wherein car demands were assumed to be stochastic. The model might useful in identifying good strategies for the size of rail - car fleets and allocation of the freight cars. Recently, in [9], the authors proposed an online preference learning algorithm (OnPL) that can dynamically adapt the policy for dispatching AGVs to changing situations in an automated container terminal. The policy is based on a pairwise preference function that can be repeatedly applied to multiple candidate jobs to sort out the best one. The paper presented that, OnPL can relearn its policy in real time, and can thus adapt to changing situations seamlessly. In comparison to OnPL, other methods cannot adapt well enough or are not applicable in real time owing to the very long computation time required. In [10] the authors proposed a method for simultaneous intrinsic and extrinsic calibration of tricycle robots equipped with a sensor like a planar laser scanner. Parameter estimation is achieved by comparing the motion expected from kinematic equations and the sensor egomotion. Two different kinematic models have been investigated: the standard tricycle model and an asymmetric model taking into account the different load conditions affecting the wheel steer axis when a real industrial AGV moves forward and backward. With this method, the experiments indicated the values computed with the proposed method are stable and yield accurate AGV navigation capabilities. In [11], some policies to avoid collision and online control strategies to deal with deadlock were proposed. They are effective both for unidirectional and bidirectional path and fully utilize space compared to traditional zone-control scheme. The proposal can be applied in systems with large number of AGVs and large-scale size of spaces with flexible path network.

2.1. Model formulation

2.1.1. Concepts of FSP (Fleet Sizing Problems)

Some detailed concepts of the FSP with time windows (FSPTW) is given by following as accepted from [12].

- *Jobs and time windows:*

A job is determined as the case of moving from one pickup and delivery point (PDP) to another by a vehicle. For each job a time window $[a_i;\ b_i]$ is combined where a_i is the release time of job i from a PDP and b_i is the due time to start job i. The value of b_i is computed from the release time of next jobs of job i and the size of the buffer. For example, with a buffer of size n (i.e. the capacity of n items) the due time for job i is calculated as: $b_i = a_{i+n}$. This means that job i should start before the release time of job $i + n$ to have at least one available slot for job $i + n$ in the buffer.

- *Compatible jobs:*

If two jobs can be done consecutively by same vehicle, they will be compatible

jobs. Jobs i and j are compatible if one vehicle can pick up job i from a pickup point P_i at a time s_i and deliver it to a delivery point D_i, and then can still travel to a pickup point P_j to pick up job j within the time window of job j. Mathematically, i and j are compatible if $s_i + tP_iD_i + tD_iP_j \in [a_j; b_j]$, where t_{PD} is the transport time from point P to point D and $[a_j; b_j]$ is the time window of job j.

- *Problem modeling:*

The FSP with time windows (VRPTW) is modeled as the Fig 1. Each node represents a pickup time for a job. Fig 1 shows an example of model the solution of one simple FSP with eight jobs (numbered from 1 to 8) and three vehicles. Vehicle 1 carries three jobs 5; 3; 4, vehicle 2 carries fours jobs 2; 1; 8; 6 and the last vehicle carries a job 7. The cell with white color presents the end of certain vehicle. By this representation, finding the optimal of the number of vehicles is becoming finding a permutation which we can include as few white cells as possible while till ensuring that the constraints are not violated. By this model, the objective is to find the permutation of the number of job to the following constraints: every job in the network must be pickup only once by one of the vehicles.

Figure 1. *Typical representation of each solution.*

2.1.2. The FSP in container terminals (CTs)

In CTs, transporting containers between two areas named quay and stack areas is concerning the circumstances of a fleet of vehicles. The quay area is the place where vessels are berthed and there are quay cranes (QCs) to unload and load containers from/to vessels. The stack area consists of a number of stack blocks served by stack cranes (SCs). Each quay or stack area has a number of PDPs for vehicles to transfer containers. Each PDP is facilitated by one crane. Once a vessel is berthed, the discharging and then loading operations are performed, usually separately.

In the discharging phase, QCs unload containers from vessels and pass them to transfer vehicles which then transport them to the stack area. If vehicles can pick up containers by themselves, QCs place the containers in the buffer, a place underneath the cranes, for vehicles to collect. Otherwise, the QCs have to wait for vehicles to come and then place the containers directly onto them. The capacity of the buffer is limited and it is required that containers should be picked up from the buffer before it gets full, i.e. there should always be at least one free slot available in the buffer. This is to minimize the expensive waiting times of QCs.

At the stack area, SCs collect containers from vehicles and place them in the stacks. After passing a container to a SC, the transfer vehicle will come back to the QCs to collect another container. The buffer for SCs is similar to the one for QCs: vehicles drop off containers in the buffer from which SCs pick them up. In this paper, it is assumed that there are always free slots available in the SC buffer for vehicles to drop off containers. After the discharging phase finishes, the loading phase will start. The loading tasks are similar to the discharging tasks but in the opposite direction. The unloading/loading process of a ship at a CT can be described as follows (see figure 2).

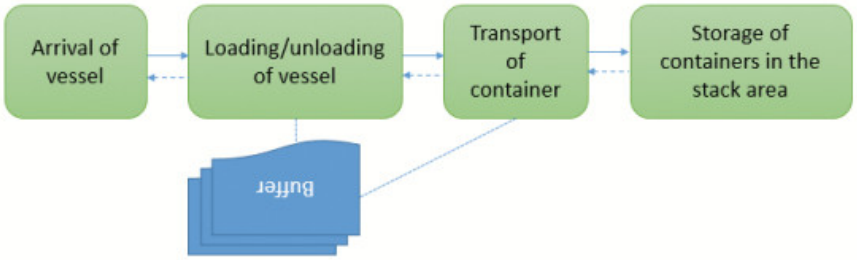

Figure 2. *The Container Terminal process.*

3. RESEARCH METHODOLOGY

This section presents an efficient method to solve FSP formulation so that the objectives are met and the constraints are satisfied. The algorithm that is adapted to solve the problem is a genetic algorithm (FSGA) that is a class of adaptive heuristics based on the drawing concept of evolution ”survival of the fitness” developed by Holland [13]. A FSGA starts with a set of individuals referred to as initial population. Each individual is a solution to the problem, and the initial population is generated either randomly or using some of heuristics. A selection operator will then be used to select the parents based on their fitness computed by evaluation function. The selected parent individuals will then be recombined via the crossover operator to create new population. The next step will be to mutate a small number of newly obtained chromosomes or using some local search in order to introduce a level of randomness that will preserve the GA from converging to a local optimum. The GA will then reiterate through this process until a predefined number of generations has been produced, or until there was no improvement in the population, or a predefined level of fitness has been reached, or ideal solution has been found.

3.1. Chromosome presentation

Figure 3. *Typical representation of each chromosome.*

Each chromosome or individual of FSPTW with GA is represented as a chain of integers, each number of chains as representing a job. The chromosome’s length is n if we have n jobs. In this string, chromosome S is a permutation of n positive integers which value between 1 and n. Fig. 3 shows a possible solution FSPTW with 10 jobs. The difficulty with this representation is how to determine the number of vehicles. In the following paragraph, we present in greater detail how to calculate fitness value. This formulation below is necessary for FSGA as it is used to determine the fitness of the solution as we called splitting procedure. Let S_i represents the pickup time for job i; y_i: represents the chosen pickup time for the next job to be done by the same vehicle as job i; n: number of jobs, $n \in N$; C_{x_i} : set of nodes that are compatible with node x_i; The objective of the VRPTW is to serve all the jobs such that the following objectives are met and the following constraints are satisfied.

Constraints:

- Time window constraints are observed $a_i \leq S_i \leq b_i$.
- Each job is pickup exactly once.
- Each job cannot be done by more than one vehicle.

Objective:

Minimize the total number of vehicles used.

By the presentation of permutation of positive integers, the second and third constraints will be satisfied through splitting function which define the fitness value of individual. This function can be split chromosome into many different partition (from begin node to end node), each partition describe the number of jobs and the processed order of jobs in one vehicle. The main objective of FSPTW now converted into how to find the permutation which supply the optimal splitting. Splitting function which can find an splitting conforms above constraints.

Algorithm 1 Splitting function returns j as evaluation fitness and $V[i]$ with $i = 1 \div j$ are set of vehicles in where each set describes each vehicle

```
for i ∈ {1, .., n} do
    V[i] = ∅
end for
j=0
for i ∈ {1, .., n} do
    if (checkTimeWindows(S[i])) then
        V[j] = V[j] ∪ S[i]
    else
        j++; V[j]=S[i];
    end if
end for
```

Algorithm 2 Check time window function with p_i is calculated as job as i index compatible jobs with job $i-1$ in permutation representation

```
if p_i < a_i then
    vehicle wait till a_i
    return true
else
    if (p_i ≥ a_i)&(p_i ≤ b_i) then
        vehicle pick up job i at p_i
        return true
    else
        return false;
    end if
end if
```

3.2. Crossover

The crossover operator is analogous to reproduction. In this more than one parent is selected and one or more offsprings are produced using the genetic material of the parents. In this paper we used crossover operator based on [14], two individuals are randomly selected from the population and the better one becomes the first parent P_1, two cutting points i and j are randomly selected in P_1. Substring $P_1(i)$, .., $P_1(j)$ is copied into $C_1(i)$, .., $C_1(j)$. P_2 is swept circularly from $j + 1$ onward to complete C with the missing nodes. C is also filled circularly from $j + 1$.

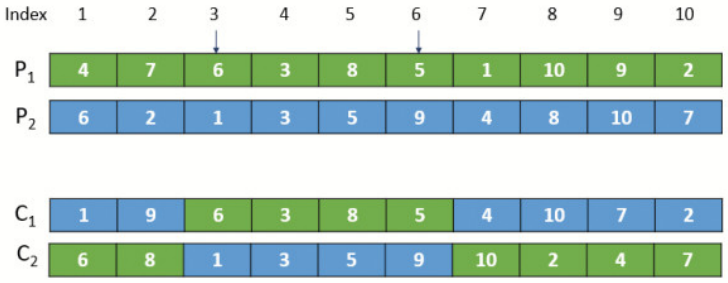

Figure 4. *An illustration of OX crossover.*

3.3. Mutation

Inversion mutation, generates two cut-points along the length of chromosome, and then reverses the material between these as figure 5.

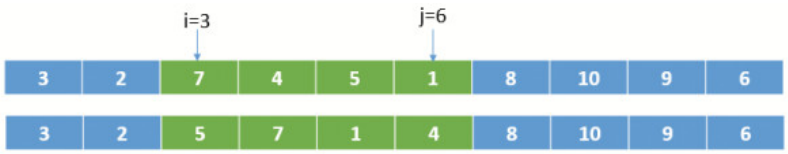

Figure 5. *An example of mutation.*

3.4. Local search

The objective is to minimize the number of fleet sizing as much as possible and even minor changes that may generate a sequence of jobs that can be carried by the same vehicle (the longest possible compatible jobs). We use a variant based on two-opt heuristic as local search [15]. A child C produced by crossover or mutation operator is then improved by local search. We consider Wider neighborhoods increase the GA running time without improving the final result. Small neighborhoods are sufficient because they are compensated by the intrinsic parallelism of the GA: indeed, the solutions which can be reached by crossovers from a given population define a kind of wide neighborhood that brings enough diversification. Ea h iteration of LS scans in $O(n^2)$ all possible pairs of distinct nodes $(u;\ v)$. These nodes may belong to the same vehicle or to different vehicle and one of them may be the depot. For each pair, the following simple moves are tested x and y are the successors of u and v in their respective trips.

Algorithm 3 2-opt exchange heuristics local search

```
for u ∈ {1, n − 2} do
    for v ∈ {count1 + 1, .., n − 1} do
        swapposition(u, v)
        if the new solution is better then
            the current then set it as the current
        end if
    end for
end for
```

3.5. Adjust Learning Method

During evolution, the algorithm might be fall into local optimal, which makes the best solution likely not to improve. For this reason, the best individual in the last generation will be adjusted as algorithm 4.

Algorithm 4 Adjusting individual after evolution

```
for all vehicles i do
    if num_of_job(i) ≤ θ then
        for all jobs j ∈ vehicle i do
            for all vehicles k ≠ i do
                if insert j into k is validated then
                    removed job j in vehicle i; insert j into k;
                end if
            end for
        end for
    end if
end for
```

3.6. The pseudo-code for FSGA

Now, it is time to put together the components of FSGA and present the pseudo-code for the whole algorithm 10. FSGA starts with the initialization randomly of

the population. It then evaluates the individuals based on the fleet size of each individual using Algorithm 1. FSGA selects individuals for the applying genetic operators for the next generation using the tournament selection. Algorithm 3 is then applied to individuals after crossover or mutation operators for possible improvement of their fitness. This loop will continue until the stopping criteria are met and the best found solution will be adjusted to return as the optimal solution.

Algorithm 5 *FSGA*

1: Initialise population P_0)
2: Evaluate population P_0
3: **while** stopping criteria not met **do**
4: Ascertain individual's fitness
5: Select one or two individual(s) from the population P_t with a probability based on fitness to participate in genetic operations (see Section).
6: Create new individual by applying genetic operations with specified probabilities (see Section).
7: Evaluate new population $P(t+1)$
8: $P_t = P(t+1)$
9: **end while**
10: return the best individual and adjust the best

V1	0	1	2	3	4	5	6	7	8	9	10	83	11	12	13	14	15	16	92	17	18	19	20
V2	21	22	23	24	25	26	27	28	29	30	31	32	84	85	33	87	34	35	36	37	38	39	
V3	40	41	42	43	44	45	46	47	48	94	49	89	50	51	52	53	54						
V4	55	56	57	58	59	60	61	62	63	64	65	66	88	67	86	68	90	69	93	70	71		
V5	72	73	74	75	76	91	77	78	79	80	81	82											
V6	95	96	97	98	99	20																	

Figure 6. *The best of solution after adjusted.*

4. EXPERIMENTAL SETTINGS

This section describes computational experiments carried out to investigate the performance of the proposed FSGA. The experimental use the test cases for this research from [1] using the real-world data from two of our European port partners. The real-world data is the number of QCs, the number of containers, the size of buffers, the distances between QCs and SCs (in [1]), etc. In the two ports (port A and port B), the maximum numbers of QCs on one vessel are three and two, respectively. 100 are the numbers of containers to be discharged which are considered to be in line with the actual transactions in the two ports.

At present, port A has six container stack blocks and port B has two blocks, and we assume containers are distributed evenly between the blocks. Each stack block is facilitated with one SC. Given the actual space under or next to the cranes, the sizes of the buffers vary from 0 to 10. We also use the assumptions that they have made in generating the test cases are as follows: 1) the speeds of vehicles are constant; 2) the processing time of quay cranes follows the given distributions (section VI-B) and 3) no waiting time of vehicles for stack cranes has been considered, similar to [1].

4.1. Time windows of jobs

As mentioned earlier in section 3.1, a container has to be picked up within a certain time window, which is the duration between the release time and due time of this container. The release time of a container depends on crane's cycle time -

the time it takes for a crane to complete moving a container from the ship to the quay side. To make the test cases as realistic as possible, we used the crane's cycle distribution time-table like [1] from a real-world scenario [16] to generate container release time. For example, there is a 5% chance that the QC would take 30-40 seconds to process a container. Given the release time of a specific container, we can then calculate its due time following the calculation in section 3.1 To do so, first we calculate all the possible pickup times for each container and identify those that are compatible with each other, we calculate the travel time of vehicles between PDPs (QCs and SCs) based on the distances between PDPs and the speeds of vehicles, which are set to be $4\ m/s$ and $2\ m/s$ for the empty and load vehicles respectively, based on common industrial specifications. Given the pickup and travel times, now we can calculate all the compatible nodes of a given node using the procedure in algorithm 1, 2.

4.2. Parameter settings of FSGA

The results presented below are based on the following parameters:

- Population size = 50
- Generation number = 200
- Crossover rate = 0.6
- Mutation rate = 0.4
- Repetition for experiments = 30

5. RESULTS AND DISCUSSION

We ran FSGA for 30 times and compared the results to that of CPLEX [1]. The FSGA also gained quite good results when compared to CPLEX: firstly, FSGA can find the same optimal fleet size as CPLEX in all runs, secondly, FSGA can only find the same optimal fleet size as CPLEX in fewer than 30 runs, thirdly, CPLEX runs out of memory but it provides an integer lower bound with no proof of optimality and the last. We coded FSGA in C++. All the experiments were conducted on a Core 2 Duo CPU 2.1 GHz with 8 GB RAM. We created 165 test cases using the settings given in Sec. 5. Experimental results are summarized into Tables I. Table I shows that FSGA significantly outperforms CPLEX in almost of test cases. CPLEX can only solve 56 cases (the smaller-scale ones) and it cannot solve 109 larger cases due to out-of-memory issues, whereas FSGA is able to solve all cases. This table also shows the effectiveness of the FSEA in terms of identifying the global optima. In 56 out of 56 cases, FSEA finds the global optima as found by CPLEX.

Table 1. *Summary of the comparison results between FSGA and CPLEX.*

Port	No. of QCs	No. of test cases	Solvable cases		FSEA found CPLEX's optima
			FSGA	CPLEX	
A	1	33	33	10	10
	2	33	33	11	11
	3	33	33	12	12
B	1	33	33	11	11
	2	33	33	12	12

6. CONCLUSIONS

In this research, we have developed a genetic algorithm able to identify the suitable number of vehicles in environments with shuttle transportation tasks (ESTTs), in both static and uncertain situations. We considered the travel time of vehicles and the processing time of machines to be the main sources of uncertainty. We compared the results of our algorithm, FSGA, with that of the commercial, state-of-the-art solver CPLEX. The results showed that FSGA was significantly better than CPLEX in most cases and especially in all large-scale cases. FSGA also has another advantage over CPLEX: while CPLEX cannot solve the FSP in certain cases, FSGA is able to solve the problems to find the robust solutions. The results also show that the proposed algorithm is able to provide the optimal robust solution in most of the cases.

REFERENCES

[1]. T. T. N. Z. Y. a. I. J. Shayan Kavakeb, *"Evolutionary fleet sizing in static and uncertain environments with shuttle transportation tasks - the case studies of container terminals,"* IEEE Computational Intelligence Magazine, vol. Volume: 11, no. Issue: 1, pp. 55 - 69, 2016 .

[2]. R. &. E. P. Arifin, *"Determination of vehicle requirements in automated guided vehicle systems: A statistical approach,"* Production Planning and Control, 2000.

[3]. R. Kasilingam, *"Mathematical modeling of the AGVS capacity requirements planning problem,"* Eng. Cost. Prod. Econ., Vols. vol. 21, no. 2, p. 171 – 175, 1991.

[4]. M. &. X. J. Ji, *"Analysis of vehicle requirements in a general automated guided vehicle system based transportation system,"* Computers & Industrial Engineering., vol. 59, pp. 544-551, 2010.

[5]. M. E. Johnson, *"Modelling empty vehicle traffic in AGVS design,"* Int. J. Prod. Res., vol. 39, no. 12, p. 2615–2633, 2001.

[6]. Q. M. a. T. Wang, *"A chance constrained programming model for short-term liner ship fleet planning problems,"* Marit. Policy Manag., vol. vol. 37, no. 4, p. 329–346, 2010.

[7]. M. M. a. N. B. ´. c, *"A fuzzy random model for rail freight car fleet sizing problem,"* Transport. Res. C-Emer., vol. 33, p. 107–133, 2013.

[8]. H. R. S. a. R. Tavakkoli-Moghaddam, *"Solving a multi periodic stochastic model of the rail–car fleet sizing by two-stage optimization formulation,"* Appl. Math. Model., vol. 34, no. 5, p. 1164–1174, 2010.

[9]. R. a. K. J. a. R. K. R. Choe, *"Online Preference Learning for Adaptive Dispatching of AGVs in an Automated Container Terminal,"* Appl. Soft Comput., vol. 38, no. C, pp. 647--660, 2016.

[10]. D. L. R. D. L. R. M. M. S. C. Fabjan Kallasi, *"A novel calibration method for industrial AGVs,"* Robotics and Autonomous Systems, vol. 94, pp. 75-88, 2017.

[11]. M.-y. Z. Z.-m. F. Ji-hong Yan, *"Scheduling Optimization for Multi-AGVs in Intelligent Assembly Workshop,"* in International Conference on Industrial Engineering and Engineering Management 2018, Hong-Kong, 2019.

[12]. R. M. B. M. D. K. a. M. W. P. S. I. F. A. Vis, *"Minimum vehicle fleet size under time-window constraints at a container terminal,"* Transp. Sci., vol. 39, no. 2, p. 249–260, 2005.

[13]. J. H. Holland, *"Genetic Algorithms and Adaptation,"* Adaptive Control of Ill-Defined Systems, pp. 317-333, 1984.

[14]. H. A. K. T. T. S. H. Ishibashi, *"Multi-objective optimization with improved genetic algorithm,"* in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, 2000.

[15]. B. G. W. Chris Groër, *"A library of local search heuristics for the vehicle routing problem,"* Mathematical Programming Computation, vol. 2, no. 2, p. 79–101, June 2010.
[16]. R. S. R. T. v. d. H. A. N. R. d. V. B. J. B. J. v. d. E. D. L. R. D. a. P. M. H. Celen, *"Famas newcon Tech. U. of Delft Erasmus U. Rotterdam",* Tech. Rep., 1997.

TÓM TẮT

SỬ DỤNG GIẢI THUẬT DI TRUYỀN GIẢI BÀI TOÁN ĐOÀN XE VỚI RÀNG BUỘC THỜI GIAN

Bài báo này trình bày một cách tiếp cận mới giải quyết cho bài toán số lượng đoàn xe với ràng buộc cửa sổ thời gian (FSPTW) bằng giải thuật di truyền. FSPTW bao gồm việc xác định số lượng phương tiện tối ưu trong môi trường công việc vận chuyển đưa đón với yêu cầu biết trước và cửa sổ thời gian được xác định trước. Những môi trường này là nghiệp vụ thiết đặt rất phổ biến với việc di chuyển đi lại các mặt hàng giữa nhiều máy bằng một đoàn xe. Ví dụ phổ biến của môi trường như vậy là nhà máy sản xuất, nhà kho và cảng container. Trong nghiên cứu này chúng tôi đã chọn hai cảng container để thí nghiệm phương pháp đề xuất. Kết quả cho thấy phương pháp đề xuất là tương đối hiệu quả, vì nó cung cấp giải pháp cạnh tranh với những phương pháp nổi tiếng nhất trong các nghiên cứu cùng lĩnh vực.

Từ khóa: Tính toán tiến hóa; Giải thuật di truyền; Bài toán đoàn xe.

Received 8th January 2020
Revised 26th February 2020
Published 6th May 2020

Author affiliation:
[1]Vietnam National University of Forestry;
[2]LeQuyDon Technical University;
[3]Haiphong Private University;
[4]National Defense Academy.
[*]Corresponding author: nguyenthihienqn@gmail.com.