An Improved Memetic Algorithm to Enhance the Sustainability and Reliability of Transport in Container Terminals

Shayan Kavakeb and Trung Thanh Nguyen

***Abstract*—In this paper, we propose an improved memetic algorithm by combining an evolutionary algorithm (EA) with Monte Carlo simulation (MCS) to identify the robust number of vehicles in environments with shuttle transport tasks (ESTTs). ESTTs are very common settings where vehicles shuttle between pickup and delivery points to transport goods. Examples of ESTTs are military bases, warehouses, manufacturing floors, and container terminals. In this study, MCS works as a local search to take into account risks of disruptions and guide the EA towards more reliable solutions. The disruptions arise from changes in the travel time of vehicles which can be caused by any breakdown, collision, accident and deadlock. Identifying the robust number of vehicles can improve sustainability and reliability of ESTTs against possible changes.**

This paper improves our previous attempts in which we studied a combination of an EA and MCS. Results of those studies showed the process of sampling in MCS is very time consuming. This prevents the EA from having an accurate estimation of the robust solutions within reasonable time. This paper improves the performance of the EA to make it possible to reach high quality solutions in reasonable time to make it practical for the real-world applications. Firstly, it proposes a new sampling technique to generate samples that reflect the worst-case scenarios better. This helps the EA to find better robust solutions using the fewer number of samples. Secondly, it proposes a new adaptive sampling technique to adjust the number of samples during evolution. To evaluate the algorithm we tested it in one of the most common ESTTs: real-world container terminals. Experimental results show that by such improvements the performance of the EA is improved significantly, making the proposed algorithm perfectly usable for its real-world case studies.

I. Introduction

The purpose of this paper is to identify the robust number of vehicles to enhance sustainability and reliability in environments with shuttle transport tasks (ESTTs). ESTTs are very common settings where vehicles shuttle between pickup and delivery points to transport goods. Examples of ESTTs are military bases, warehouses, manufacturing floors, and container terminals.

This paper extends our previous attempts [1, 2] in terms of the performance of the evolutionary algorithm (EA) and quality of solutions. In [1], we proposed a memetic algorithm by combining an EA with Monte Carlo simulation (MCS), named Fleet Sizing Evolutionary Algorithm (FSEA), to identify the robust number of vehicles in environments where vehicles shuttle between pickup and delivery points to transport goods. Examples of such environments are manufacturing factories, warehouses and container terminals. In [2], we proposed an extension on FSEA to improve its performance in terms of computational time and finding robust solutions. In this paper, to evaluate the performance of the algorithm, we choose container terminals, one of the most common and important ESTTs, as case studies. Note that although the experiments are done in container terminal, the application of the algorithm can be applied to any other ESTT with the same characteristics. The next paragraph will highlight the significance of container terminals and the importance of improving sustainability in container terminals using EAs.

In recent decades, by the increase of containerisation, container terminals has been dealing with a considerable high number of containers. For example, a small-medium size container terminal deals averagely with one million twenty-foot equivalent units (TEUs) per year [3]. Therefore, to provide competitive services by container terminals, sustainability has become a key factor. In other words, the robustness of container terminals against unexpected changes has become more important to guarantee acceptable services by container terminals. One of the key factors to maintain the sustainability of container terminals is the optimal equipment settings. This is because equipping container terminals with the robust number of equipment can prevent possible risks of changes due to uncertainties. One of the most important equipment in container terminals is transfer vehicles. These vehicles transport containers between quay and stack areas, hence the total throughput of container terminals is highly dependent on the performance of transfer vehicles. To identify the robust number of vehicles, the possible source of uncertainties should be identified and taken into account. In [1, 2], uncertainties in the travel time of vehicles were considered to be the main source of uncertainties which have a significant impact on the optimal number of vehicles. The travel time of vehicles in container terminals is under the risk of being unexpectedly changed due to many disruption factors such as breakdowns, deadlocks and collisions.

In [1, 2], to encapsulate uncertainties in solutions, FSEA used MCS to evaluate the robustness of solutions during evolution. To do so, whenever FSEA wants to evaluate the fitness of one solution it replicates n samples of that particular solution by incorporating uncertainties in the travel time of vehicles. The fitness of that particular solution is then calculated

Corresponding author: Trung Thanh Nguyen, email address: T.T.Nguyen@ljmu.ac.uk, Tel: +44 151 231 2006

S. Kavakeb and T.T. Nguyen are with The School of Engineering, Technology and Maritime Operation, Liverpool John Moores University, L3 3AF, United Kingdom; emails: S.Kavakeb@2011.ljmu.ac.uk and T.T.Nguyen@ljmu.ac.uk.

based on its robustness on those samples. The robustness of a solution is context-dependent. In many real-world situations practitioners may choose to have a robust worst-case scenario to absolutely avoid risks. In other situations, they may want to have a robust average value or a robust mode value. If a worst-case scenario is desired, the worst performance of the solution on the samples is considered the fitness of the solution so that the EA can evolve to find a solution with the better worst-case scenario; otherwise, based on certain performance measures [2] the results of evaluation on n samples should produce the fitness of the solution.

The worst-case scenario is the focus of this paper. Normally, the higher the number of samples, the more accurate the evaluation of solutions. As a result, the number of samples should be considerably high to make sure that the worst-cases are included in the samples. Results of [1, 2] show that the process of sampling in FSEA, however, is very time consuming and having a large number of samples can worsen the performance of FSEA in terms of computational time significantly. The challenge is how to improve the performance of FSEA to have high quality solutions in shorter time.

This paper attempts to resolve the above challenge. Firstly, it proposes a new sampling technique to generate samples that reflect the worst-case scenarios better. Using this technique, the EA can achieve better solutions with the fewer number of samples. Secondly, it proposes an adaptive approach to increase the number of samples depending on the convergence of FSEA. This can help FSEA to spend less time on solutions in the earlier generations where it has not converged yet and the quality of solutions is poor.

II. Related Literature

The optimal fleet size has a significant impact on the productivity of ports [4]. Having too few vehicles is not efficient and would increase the waiting time of cranes. In contrast, having too many vehicles is expensive and may increase collisions and deadlocks. However, in spite of such an important role of the optimal fleet size, the fleet sizing problem (FSP) has not received enough attention from the research community. Below is a brief review of research that studied the FSP in ports.

Vis *et al* [5] modelled the FSP as a minimum flow problem. They proposed a polynomial algorithm to solve the minimum flow problem. This research, however, has a limitation that it cannot be generalized for the cases where there is a special area (called buffer) for cranes and vehicles to temporarily store containers. Research in [5] was followed in [4] to be able to consider buffers of containers. Vis *et al* [4] modelled the FSP as an integer programming (IP) problem. The authors used the commercial solver CPLEX to solve the integer programming. Due to the limitation of the IP solvers, this model can be used only for small-case problems, for example the size of buffer and the number of containers should be considered very small. Koo *et al* [6] proposed a two-phase algorithm to identify the optimum number of vehicles in Busan port. In the proposed algorithm, the fleet size is first estimated by a heuristic. A tabu search algorithm is then applied to the given fleet size to identify the optimal routes for the vehicles. If such routes can be found the problem is solved; otherwise it increases the fleet size until it finds feasible routes.

The FSP in ports, similar to other optimisation problems [7, 8, 9, 10, 11], is subject to uncertainties in environments [1, 2]. Examples of such uncertainties are changes in the travel time of vehicles and process time of cranes. Results of [1] confirmed that the impact of uncertainties on the optimal number of vehicles is significant. In the literature there exists papers that considered uncertainties in the environment to improve sustainability of systems. In [12, 13, 14, 15], genetic algorithms and Monte Carol simulation were combined to take into account the reliability of system components. In these studies Monte Carlo simulation was used to evaluate the fitness of chromosomes by estimating the reliability of the system and maintenance status of system components. Sörensen and Sevaux [16] proposed a practical approach to combine Monte Carlo simulation with meta-heuristics for flexible vehicle routing problem. It used Monte Carlo simulation to estimate the robustness or flexibility of solutions. Sevaux and Sörensen [17] modified the genetic algorithm to compute robust machine schedules given uncertainties in the system. For this genetic algorithm new robustness measures were defined to evaluate solutions based on the robustness and distance to the baseline solutions.

As one can see, none of the above papers considered uncertainties in container terminals which leaves an important gap behind. This paper tries to close this gap by improving the performance of FSEA to identify the robust number of vehicles in container terminals under uncertainties in the travel time of vehicles. Within this study, we focus on the case of the intelligent autonomous vehicles (IAVs), a new generation of automatic vehicles, but the results can be applied to other similar type of vehicles as well.

III. An EA to Identify the Robust Number of IAVs

This section explains the EA, proposed in [1, 2], to identify the robust number of IAVs. The figures and pseudo-codes in this section are adapted from [1, 2].

A. IAVs in Ports

In ports, vehicles transport containers between quay side and stacking areas. A quay side area is a place where vessels are berthed and a stacking area is a place where containers are stacked for temporary periods. Stacking areas consist of a number of blocks to stack containers. Each block is served by a number of stacking cranes (SCs) to stack/unstack containers. Once a vessel is berthed a number of quay cranes (QCs) would be assigned to that vessel. QCs discharge containers from the vessel, vehicles then would come to collect containers and transport them to the stacking area. If vehicles can pick up containers by themselves QCs can place containers in "buffer" from which vehicles can collect containers; otherwise QCs must wait until vehicles arrive. The IAV is one of the few vehicles that is able to pick up/drop off containers by themselves thanks to a special table-shape structure named the "cassette". Therefore, they can be used in combination

with buffers in ports so that cranes and IAVs do not have to wait for each other. This way, the waiting time of both cranes and IAVs can be minimised. IAVs then transport containers to the stacking area and drop off containers in a buffer next to a SC. The SC then can come and collect containers from the buffer. Once all the containers are discharged from the vessel the loading tasks start. The loading tasks are similar to the discharging tasks but in an opposite direction.

B. Time Window for Each Container

A time window is a closed interval bracketed by the time a container becomes available in a buffer (release time) and the latest time this container must be picked up from the buffer by IAVs (due time). Each container has a time window associated to itself. Each container needs to be picked up within its time window. To help define the precise pickup time, each time window is discretized into a number of possible pickup moments [4].

C. A Graph Model for the FSP

The FSP in [4] was modelled as a directed graph. This graph shows all possible ways that IAVs can collect containers and all pairs of containers that are compatible (i.e. the two containers can be collected by the same vehicle within their time windows). Each node of this graph represents a container in one of its pickup time and each arc of this graph connects two compatible containers (nodes). This graph also has a sink node and a source node. These nodes are used to show the starting and ending points of each path. A path starts from the source node and passes through some compatible containers (nodes) to end at the sink node. Each path is actually a sequence of containers to be collected by an IAV. The objective of the problem is to find the minimum number of paths covering all containers subject to the following constraints:

1) each container can only be picked up or dropped off once
2) each container cannot be picked up or dropped off by more than one vehicle.

The number of paths is equivalent to the number of IAVs. Figure III.1 shows the graph model of an FSP example. In this example, there are three containers for IAVs to collect. Container 1 has two possible pickup times i.e. container 1 can be picked up either in its first pickup time (j_{11}) or its second pickup time (j_{12}). Similarly, container 3 has two possible pickup times: j_{31} and j_{32}. Container 2 has one start time (j_{21}). The compatible nodes are connected by directed arcs i.e. the containers that can be transported sequentially by one IVA. This figure shows one solution of this example with two IAVs. Container 1 at pickup time j_{11} and container 3 at pickup time j_{32} are assigned to IAV 1 and container 2 is assigned to IAV 2. Thus, the fleet size for this particular example is two.

D. Representation

The FSP in FSEA is represented as a string of pairs $< x_i, y_i >$, $1 \leq i \leq n$ where n is the number of containers; x_i represents index of the pickup time of container i; and y_i represents the container to be transported by the same IAV after container i.

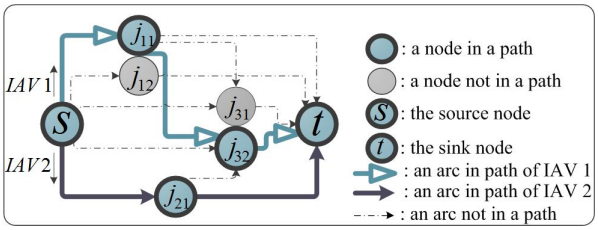

Figure III.1. An example of how an FSP can be modelled as a directed graph.

E. Recombination

To improve the fitness of an individual a new recombination-based heuristic operator was developed in [1]. At each generation, FSEA applies the heuristic to individuals to improve their fitness by reducing the number of IAVs in each individual. To do so, for each individual it selects one IAV randomly to remove, let us call it *IAV_delete*. It then tries to assign all containers of IAV_delete, if possible, to other IAVs. By removing all the containers of IAV_delete, this IAV can be removed from the fleet (Figure III.2).

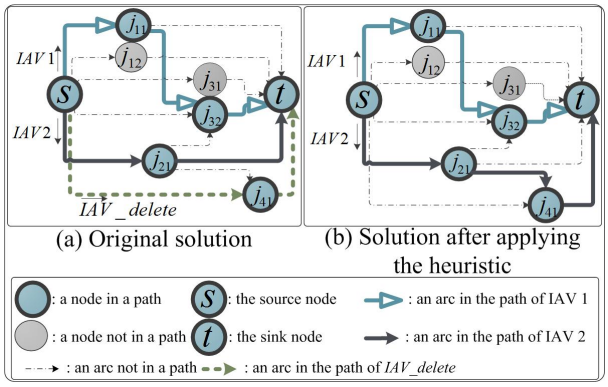

Figure III.2. This figure shows an example of how the heuristic reduces the number of IAVs. The left plot shows the original solution with three IAVs. The right plot shows the solution after being improved by the heuristic. As can be seen, by moving container 4 (j_{41}) to the list of containers of IAV 2, the fleet size decreases to two.

F. Mutation

The mutation operator uses a similar idea to the heuristic. The purpose of applying the mutation operator is to help the heuristic to remove IAV_delete. It moves containers between IAVs apart from IAV_delete in a hope that it can make room to insert containers from IAV_delete (Figure III.3).

G. Evaluation

In the static case i.e. no uncertainty, the number of IAVs (i.e. the number of paths in the graph model of the FSP) in a solution can be considered the fitness of that solution. In the uncertain case, however, evaluating the fitness of individuals only based on the number of IAVs may not be totally realistic. There may be a case where individuals have the same number of IAVs but have different sequences of containers i.e. they have different schedules for the same number of IAVs. Those schedules may perform differently under uncertainties and

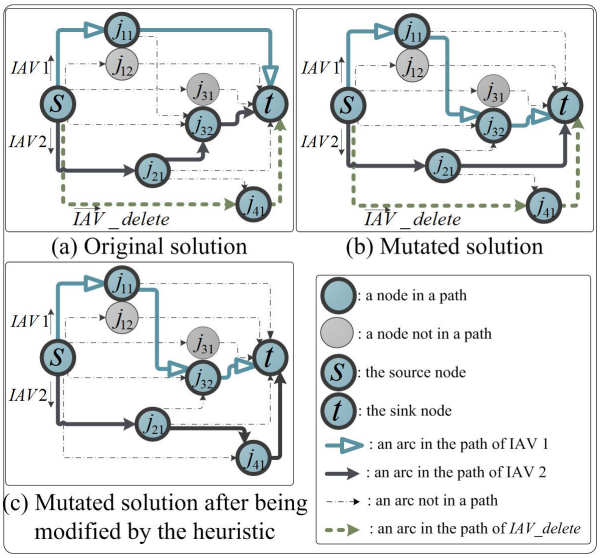

Figure III.3. An example of how the mutation can help the heuristic to remove IAV_delete. Plot a shows the original solution before being mutated in which IAV_delete has only container 4 (j_{41}). However, at this moment there is no available position for container 4 in other IAVs. To improve the situation, the mutation operator moves container 3 (j_{32}) from IAV 2 to the list of containers of IAV 1 (plot b). The heuristic then assigns container 4 to IAV 2 and removes IAV_delete from the list of IAVs (plot c).

some schedules may be more robust against uncertainties. Thus, to handle uncertainties FSEA calculates the fitness of each individual based on not only the number of IAVs in the individual but also on the robustness of the schedule of that particular solution. To do so, FSEA uses a Monte Carlo simulation (MCS) to evaluate the robustness of individuals by simulating possible uncertainties in the travel time (caused by failures such as breakdown, congestion or delay etc) that may happen to the schedules of individuals.

To measure the robustness of each solution, MCS produces a number of instance of that particular solution, each with a different failure outcome due to uncertainties. Each instance is generated by estimating possible failures of the IAVs in the solution. Those possible failures give an estimate of the time that IAVs may become unavailable due to the failures. Once a failure happens to an IAV it takes a mean-time-to-repair duration (MTTR) to get repaired and back to the system (Algorithm 1). IAVs that are in failures cannot transport their assigned containers. Those containers, if possible, should be assigned to other available IAVs to prevent causing any delay to the system. If no IAV is available, *additional IAVs* must be added to the fleet to take over those containers.

The total fleet size including the additional IAVs is used to measure the robustness of solutions. The higher the number of additional IAVs, the less robust a solution under uncertainties. Results of robustness of a solution on n samples is considered fitness of that individual. In [1], the average of total fleet size of n samples are used to calculate the fitness of individuals. In [2], we studied the impact of different robustness measures by using different aggregation functions (e.g. the maximum, minimum and mode values of the total fleet size). In this paper, because the worst-case scenario is the focus, we use

Algorithm 1 EstimateFailures

1: $F := 0$
2: $t := 0$
3: **while** $t < makespan$
4: Generate a random exponential value t_e using the parameter λ
5: $t_f := t_e + t$
6: $t_r := t_f + MTTR$
7: **if** $t_f < makespan$
8: $F := F \cup \{< t_f, t_r >\}$
9: $t := t_r$
10: **return** F

where F is the set of duration of failures, t is the simulation time, $makespan$ is the time that the last job is done, λ is the failure rate of IAVs and $MTTR$ is the mean time to repair.

the maximum function, i.e. the largest total fleet size over n samples, as the fitness of individuals. To get a more accurate evaluation on the robustness of a solution, we need to use more Monte Carlo samples, i.e. we need to generate more instances of the solution under uncertainties. The results of [1], however, showed that the process of sampling is very time consuming and in large-scale problems almost makes it impossible to have an accurate estimation of robust solutions. In this paper, Section IV, we propose two new extensions on FSEA to have a better estimation of fitness of individuals in reasonable time using the fewer number of samples.

H. Dynamic Sampling Technique

In [2], we proposed an extension on FSEA, named iFSEA, to improve the performance of FSEA. It uses a dynamic sampling strategy to increase the number of samples step by step. It starts with a small number of samples to evaluate solutions and step by step it increases the number of samples after a certain number of generations until it reaches its maximum. This technique is improved in this paper to an adaptive approach in Section IV.

IV. Extensions on FSEA

This section discusses two extensions on FSEA, called FSEA+. We first explain an approach to generate the samples that reflect the worst-case scenarios better. We then explain an adaptive sampling technique to adjust the number samples.

A. Generating Samples that Reflect the Worst-case Scenarios Better

As mentioned in Section I, the purpose of this paper is to identify the robust number of IAVs that has the best performance in the worst-case scenario. In order to find such a robust solution, the number of samples should be considerably high to be confident that the worst-case samples are included in the generated samples. Results of [1], however, showed the process of sampling in FSEA is very time consuming and increase in the number samples deteriorates the performance of FSEA significantly, so such increase in the number of samples is not possible. Thus, we propose a new approach that generates the samples that reflect the worst-cases better rather than increasing the number of samples.

Recall from Section III, we encapsulate possible uncertainties in an individual by generating multiple instances (called samples) of the same individual, of which in each sample we introduce various failures to the IAVs. The failures for IAVs are estimated using an exponential distribution with the parameter λ - the failure rate of IAVs. The total time that vehicles are under repair is called the duration of failures. Due to uncertainties, different samples of the same individual may have different duration of failures. Furthermore, when we compare two different samples of the same individual, it can be assumed that the sample with a larger total duration of failures is more likely to need more additional vehicles to cover the failures, and hence to have a larger fleet size (i.e. worse fitness). If this assumption is true, then samples with the largest durations of failures might be the ones with the largest fleet size, i.e. they are the worst-case scenarios.

Following this assumption, to find the worst-case scenarios, instead of simulating uncertainties using MCS in all possible samples of an individual, we propose to just apply MCS to those samples with the largest total durations of failures. This will save computational time because MCS is very time consuming while estimating the duration of failures is computationally cheap.

In this paper, our proposed idea above is implemented in a sampling procedure named MC+. To do so, in the process of sampling, for each individual, MC+ generates a pool of samples with a considerably large size m. For each sample, MC+ does not apply MCS directly but only estimates the total duration of failures given uncertainties (based on an exponential distribution). Among those samples it selects n ($n < m$) samples with the largest total durations of failures and evaluates the robustness on those n samples (Algorithm 2).

Algorithm 2 MCS+

1: Identify FS, the number of IAVs in individual $\vec{X}$
2: F := 0 //F is the set of duration of failures
3: FSL := 0
4: $sampNo$:= *GetNumberOf*Samples()
5: **for** j **from** 1 **to** m //m is the size of the pool of samples
6: **for** i **from** 1 **to** FS
7: $F[j][i]$:= *EstimateFailures*()
8: Sort F based on the total duration of failures in each sample
9: **for** j **from** 1 **to** $sampNo$
10: UJ : = 0 //UJ is the list of uncovered containers
11: $tempFS$: = FS
12: **for** i **from** 1 **to** $tempFS$
13: Identify the uncovered containers due to IAV i's failures in $DF[j]$ and add them to UJ
14: **for** i **from** 1 **to** *len*gth(UJ)
15: **if** container $UJ[i]$ can be covered by an available IAV k
16: Assign container $UJ[i]$ to IAV k
17: **else**
18: $tempFS$:= $tempFS$ + 1; //add a new IAV to the fleet
19: Assign container $UJ[i]$ to the newly added IAV
20: $FSL := FSL \cup \{tempFS\}$
21: **return** the maximum of FSL

where FSL stores the results of evaluation of individual $\vec{X}$ on the samples, $sampNo$ is the number of samples for each evaluation, $F[j][i]$ is the duration of failures of IAV i in sample j and $tempFS$ is the fleet size for the current sample.

B. Adaptive Sampling Technique

FSEA has a disadvantage: it applies the same large number of MC samples to every individual in every generation, regardless of the quality of the individuals. This is not effective because any samples on poor-quality individuals will be wasted. To improve this situation, we propose an adaptive sampling technique in FSEA+. This technique evaluates the robustness of solutions in the earlier generations using fewer number samples and it increases the number of samples adaptively based on the convergence of the population. The idea is that, if the algorithm finds a (local or global) optimum, we need to accurately investigate the impact of uncertainties on that optimum. To do so, we will look for sign of (temporary) convergence and whenever it happens we will increase the number of allowed samples for the MCS. We consider the algorithm as temporarily converged if after α consecutive generations we do not observe either of the following criteria in the best individual: 1) a decrease in the fleet size; 2) a decrease in the number of containers assigned to IAV_delete, which is the vehicle that the recombination heuristics (see Subsection III-E) wants to eliminate.

Starting from an initial number of samples n_0, whenever a temporary convergence occurs according to the criteria above, the algorithm increases the number of samples by a value β. The algorithm follows this approach until it reaches the maximum allowed number of samples n. Algorithms 3 and 4 show the pseudo-codes for the adaptive sampling technique and FSEA+, respectively. As mentioned in Subsection III-H, our previous algorithm iFSEA also has a dynamic approach to adjust the number of samples during evolution. That approach, however, is not adaptive: it always increases the number of samples after each fixed number of generations regardless of the convergence of the algorithm. That approach is also problem-dependent and for each problem its parameters should be tuned.

Algorithm 3 GetNumberOfSamples

1: **if** no improvement happens to the populations for α consecutive generations
2: **if** $sampNo$ + β > n //β is the step to increase the number of samples
3: **return** n //n is the maximum number of samples
4: $sampNo$:= $sampNo$ + β //$sampNo$ is the number of samples
5: **return** $sampNo$
6: **else**
7: **return** $sampNo$

Algorithm 4 FSEA+

1: Initialize population P_t
2: **Evaluate** population P_t by *MCS+*()
3: **for** $genCounter$ **from** 1 **to** $genNo$
4: **Select** elements from P_t to copy into P_{t+1}
5: **for** i **from** 1 **to** $popSize$
6: Apply the mutation operator to individual i
7: Apply the heuristic operator to individual i
8: **Evaluate** new population P_{t+1} by *MCS+*()
9: P_t := P_{t+1}
10: **return** the best individual

where $genNo$ is the maximum number of generations and $popSize$ is the size of the population.

V. Experimental Results

This section provides results of the experimental study. It first provides details of test cases and parameters settings. It then shows the improvement in robust solutions by comparing results of FSEA+ and iFSEA for the worst-case scenarios. Following that, it discusses improvement in the performance of the algorithm by comparing results of non-adaptive FSEA+ and adaptive FSEA+. Non-adaptive FSEA+ is the case where the number of samples during the evolution is constant i.e. $n_0 = n$ and adaptive FSEA+ adjusts the number of samples adaptively.

A. Test Cases and Parameter Settings

We consider a European port the case study of this paper[1]. All settings are from real-world data of this port. In this port there are three QCs available to discharge/load containers from/into vessels, so we vary the number of QCs from one to three. There are six stacking blocks available to stack containers each is served by one SC. We assume that the containers are distributed evenly between the stacking blocks. The size of the buffer is varied from 0 to 10. The speeds of loaded and empty IAVs are considered 2 m/s and 4 m/s, respectively. The number of containers to be discharged is 100. The distances between the QCs and SCs are as in [1, 2]. The parameter setting for the experiments are shown in Table I. Note that due to the lack of actual failure rates and MTTR of IAVs, we chose the same failure rate and MTTR as in [18].

Table I
Parameters setting for FSEA, iFSEA and FSEA+

Alg.	Parameter	Value	Description
FSEA+	n_0	60	Initial number of samples
	β	20	Step to increase the number samples
	α	5	Temporary convergence criterion
	m	3000	Size of the pool of samples
FSEA, iF-SEA & FSEA+	n	100	Maximum number of samples
	$genNo$	100	Maximum number of generations
	$popSize$	15	Size of the population
	λ(failures/s)	1.0×10^{-3}	Failure rate of IAVs
	MTTR(s)	500	Mean time to repair of IAVs
	other	as in ([1, 2])	

B. Comparing the Quality of Robust Solutions

Results of FSEA+ and iFSEA on the test cases are shown in Figure V.1. The test cases are categorised based on the number of QCs that are involved in discharging tasks. The results show that in 26 out of 33 cases FSEA+ found new worst-case solutions with larger fleet sizes than what iFSEA found. This confirms that using the new sampling technique can help FSEA+ to find better robust solutions.

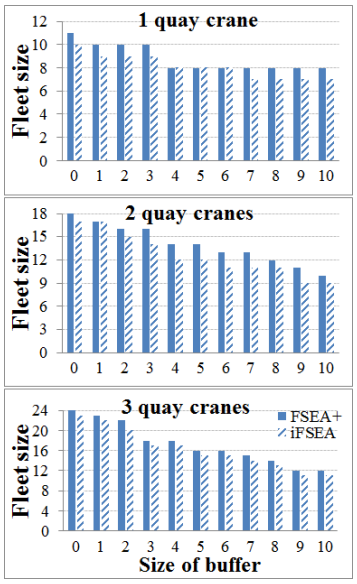

Figure V.1. Robust solutions for the worst-case scenarios by 100 samples per evaluation.

C. Process Time Comparison

In this subsection, we first investigate how FSEA+ improves the process time in comparison to the previous versions: iFSEA and FSEA. To do so, we calculate the amount of time that each algorithm needs to find the best robust solutions in the worst-case scenarios. To find the same worst-case-scenario robust solutions as FSEA+, FSEA and iFSEA have to use 3000 samples per evaluations. This causes these algorithms to spend a large amount of time. Due to these two algorithms taking very long time to reach the robust solutions of the same quality as those of FSEA+, at the submission date we have only been able to complete the experiments of iFSEA/FSEA with 3000 samples per evaluation in one test case: three QCs and buffer size equals six. For this case, process times of FSEA/iFSEA and FSEA+ are 62,555 and 2,800 seconds, respectively. Note that although the other experiments have not completed yet, the difference in comparison to FSEA+ should be as significant as above.

To evaluate the contribution of the new adaptive approach (Subsection. IV-A) to improve the performance of FSEA+, we compare the adaptive FSEA+ with the non-adaptive one. Figure V.2 shows that the new adaptive sampling technique significantly reduces the process time of FSEA+.

VI. Conclusion

In this paper, we extend our attempts in [1, 2] to identify the robust number of IAVs in ports. The extensions are: 1) generating the samples that reflect the worst-case scenarios better; 2) proposing an adaptive sampling approach to increase

[1]Due to the confidential agreements we cannot reveal the identity of this port.

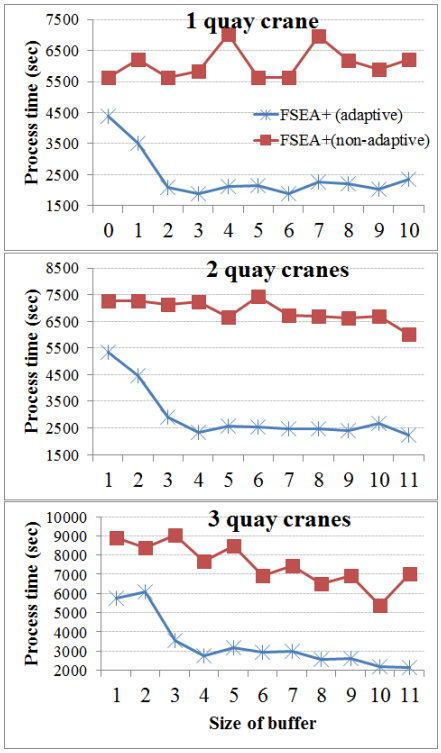

Figure V.2. This figure shows the improvement in the process time of FSEA+ following the adaptive sampling approach.

the number samples based on the convergence of the algorithm. Experimental results show that by using such sampling technique, better robust solutions can be achieved within fewer number of samples. This help improve the reliability and sustainability of vehicle fleets in ports. Moreover, using the new adaptive approach, the performance of the algorithm is increased significantly.

Acknowledgment

This research was supported by a research grant from RCUK NEMODE – New Economic Models in the Digital Economy, Network+, a European project named Intelligent Transportation for Dynamic Environment (InTraDE) and a Seed-corn funding grant by the Chartered Institute of Logistics and Transport.

References

[1] S. Kavakeb, T. T. Nguyen, Z. Yang, and I.Jenkinson, "Evolutionary fleet sizing in environments with shuttle transportation tasks - case studies of container ports," *Submitted to IEEE Computational Intelligence Magazine*, 2014.

[2] S. Kavakeb, T. T. Nguyen, Z. Yang, and I. Jenkinson, "Identifying the robust number of intelligent autonomous vehicles in container terminals," in *EvoSTOC, EvoStar*, 2014.

[3] U. N. C. on Trade and Development, "Review of maritme transport," 2013.

[4] I. F. A. Vis, R. M. B. M. De Koster, and M. W. P. Savelsbergh, "Minimum vehicle fleet size under time-window constraints at a container terminal." *Transportation Science*, vol. 39, no. 2, pp. 249 – 260, 2005.

[5] I. F. A. Vis, R. M. B. M. De Koster, K. J. Roodbergen, and L. W. P. Peeters, "Determination of the number of automated guided vehicles required at a semi-automated container terminal," *Journal of the Operational Research Society*, vol. 52, no. 4, pp. pp. 409–417, 2001.

[6] P. Koo, W. Lee, and D. Jang, "Fleet sizing and vehicle routing for container transportation in a static environment." *OR Spectrum*, vol. 26, no. 2, pp. 193 – 209, 2004.

[7] H.-G. Beyer and B. Sendhoff, "Robust optimization–a comprehensive survey," *C. M. in applied mechanics and engineering*, vol. 196, no. 33, pp. 3190–3218, 2007.

[8] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transaction on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[9] T. T. Nguyen, "Continuous dynamic optimisation using evolutionary algorithms," Ph.D. dissertation, School of Computer Science, University of Birmingham, http://etheses.bham.ac.uk/1296, January 2011.

[10] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[11] F. Neri, G. L. Cascella, N. Salvalore, and S. Stasi, "An adaptive prudent-daring evolutionary algorithm for noise handling in on-line pmsm drive design," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. IEEE, 2007, pp. 584–591.

[12] M. Marseguerra and E. Zio, "Optimising maintenance and repair policies via a combination of genetic algorithms and Monte Carlo simulation," *Reliability Engineering & System Safety*, vol. 68, no. 1, pp. 69–83, 2000.

[13] M.Marseguerra, E. Zio, and L. Podofillini, "Condition-based maintenance optimization by means of genetic algorithms and Monte Carlo simulation," *Reliability Engineering & System Safety*, vol. 77, no. 2, pp. 151–165, 2002.

[14] M. Marseguerra, E. Zio, and L. Podofillini, "Genetic algorithms and Monte Carlo simulation for the optimization of system design and operation," in *Computational Intelligence in Reliability Engineering*. Springer, 2007, pp. 101–150.

[15] M. Cantoni, M. Marseguerra, and E. Zio, "Genetic algorithms and Monte Carlo simulation for optimal plant design," *Reliability Engineering & System Safety*, vol. 68, no. 1, pp. 29–38, 2000.

[16] K. Sörensen and M. Sevaux, "A practical approach for robust and flexible vehicle routing using metaheuristics and Monte Carlo sampling," *Journal of Mathematical Modelling and Algorithms*, vol. 8, no. 4, pp. 387–407, 2009.

[17] M. Sevaux and K. Sörensen, "A genetic algorithm for robust schedules in a one-machine environment with ready times and due dates," *Quarterly Journal of the Belgian,*

French and Italian Operations Research Societies, vol. 2, no. 2, pp. 129–147, 2004.

[18] B. Farling, C. Mosier, and F. Mahmoodi, "Analysis of automated guided vehicle configurations in flexible manufacturing systems," *International Journal of Production Research*, vol. 39, no. 18, pp. 4239–4260, 2001.