Transformer Surrogate Genetic Programming for Dynamic Container Port Truck Dispatching

Xinan Chen$^{1[0000-0001-9620-3264]}$, Jing Dong$^{2,3[0000-0001-9184-2622]}$, Rong Qu$^{3[0000-0001-8318-7509]}$, and Ruibin Bai$^{3[0000-0003-1722-568X]}$

[1] School of Computer Science, University of Nottingham Ningbo China, Ningbo, China {xinan.chen,ruibin.bai}@nottingham.edu.cn
[2] Department of Engineering, University of Cambridge, Cambridge, UK jd704@cam.ac.uk
[3] School of Computer Science, University of Nottingham, Nottingham, UK rong.qu@nottingham.edu.cn

Abstract. In the wake of burgeoning demands on port logistics, optimizing the operational efficiency of container ports has become a compelling necessity. A critical facet of this efficiency lies in practical truck dispatching systems. Although effective, traditional Genetic Programming (GP) techniques suffer from computational inefficiencies, particularly during the fitness evaluation stage. This inefficiency arises from the need to simulate each new individual in the population, a process that neither fully leverages the computational resources nor utilizes the acquired knowledge about the evolving GP structures and their corresponding fitness values. This paper introduces a novel Transformer-Surrogate Genetic Programming (TSGP) approach to address these limitations. The methodology harnesses the accumulated knowledge during fitness calculations to train a transformer model as a surrogate evaluator. This surrogate model obviates the need for individual simulations, thereby substantially reducing the algorithmic training time. Furthermore, the trained transformer model can be repurposed to generate superior initial populations for GPs, leading to enhanced performance. Our approach synergizes the computational advantages of transformer models with the search capabilities of GPs, presenting a significant advance in the quest for optimized truck dispatching in dynamic container port settings. This work improves the efficiency of Genetic Programming and opens new avenues for leveraging GP in scenarios with substantial computational constraints.

Keywords: Evolutionary Algorithm · Truck Dispatching · Dynamic Optimization · Deep Neural Network · Machine Learning.

1 Introduction

In the context of the modern global economy, container ports serve as critical nodes in the logistical chain, linking various modes of transportation and facilitating the smooth transfer of goods. As international trade volumes continue to rise, the throughput capacity of container ports faces an unprecedented strain, necessitating the development of innovative optimization techniques to bolster operational efficiency [13].

One crucial area of operational efficiency pertains to dispatching trucks within the port. Efficient truck dispatching mechanisms can significantly reduce lead times, minimize fuel consumption, and mitigate the risk of congestion, thereby elevating the overall performance of the port [6]. Many optimization techniques, ranging from linear programming to metaheuristic algorithms, have been proposed to address this complex task.

Genetic Programming (GP), an evolutionary algorithm, has shown promising results in tackling the truck dispatching problem. However, a prominent bottleneck in the practical applicability of GP is the computational overhead incurred during the fitness evaluation stage. Traditional GP algorithms require each newly generated individual in the population to undergo a simulation to calculate its fitness. Not only does this process incur a high computational cost, but it also fails to exploit valuable knowledge acquired during previous algorithm iterations.

This paper aims to mitigate the limitations above by unveiling a novel Transformer Surrogate Genetic Programming (TSGP) framework. The cornerstone of this approach lies in integrating a transformer model trained to act as a surrogate evaluator for the fitness function. This innovative methodology significantly ameliorates the computational time complexity associated with the training phase by obviating the need for resource-intensive simulations. Consequently, our proposed TSGP framework demonstrates faster convergence and superior performance outcomes within the same computational time frame. Detailed experimental comparisons substantiating these claims are presented in Section 5.

In addition, the transformer model can be concurrently trained with GP algorithms, utilizing the data generated during the fitness evaluation of GP individuals. This concurrent training process eliminates additional time investment for training the transformer. Notably, the transformer leverages insights from GP individuals' structural attributes and corresponding performance metrics during their fitness evaluation. This enhances the algorithm's efficiency and contributes to a more effective fitness evaluation process. Furthermore, the trained transformer model can be strategically employed to generate superior initial populations for subsequent GP algorithms, thereby augmenting their overall performance.

The primary contributions of this paper can be summarized as follows:

- Introduction of a novel TSGP framework designed to optimize truck dispatching in dynamic container port settings.
- Deployment of a transformer model trained as a surrogate evaluator for the fitness function, significantly reducing the computational overhead associated with traditional Genetic Programming methods.
- Empirical validation demonstrating that the TSGP methodology converges faster and yields superior results within the same computational time frame.

– Illustration of the reusability of the trained transformer model for seeding improved initial populations in Genetic Programming, leading to enhanced algorithmic performance.

The remainder of this paper is organized as follows: Section 2 introduces the background of the container port truck dispatching problem and reviews the related work in port logistics optimization. Section 3 details the problem and the proposed methodology. Section 4 provides an empirical evaluation of the proposed technique, including comparisons with traditional GP and other state-of-the-art methods. Finally, Section 5 concludes the paper and outlines directions for future research.

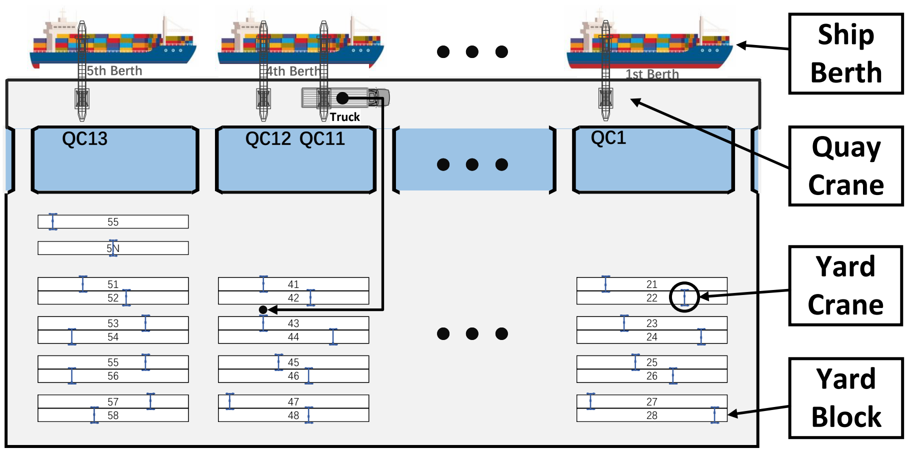

Fig. 1. Typical Layout of a Container Port

2 Background and Lecture Review

Container ports serve as essential nodes within the intricate web of international transportation networks, functioning within a globally interconnected infrastructure to facilitate the smooth and efficient transference of goods [2]. A prototypical container port shown in Fig. 1 is characterized by several critical operational elements, including ship berths, quay cranes (QCs), yard cranes (YCs), yard blocks, and a fleet of internal trucks. These disparate yet interconnected components collaborate synergistically to manage various logistical tasks, from loading and unloading containers to their subsequent intra-port transfer [24].

Truck dispatching – allocating trucks for internal container transport – plays a pivotal role in the varied operations. Efficient truck dispatching is essential for a myriad of reasons. Firstly, trucks play a vital role in the last-mile delivery of goods, linking the port to inland distribution points. Therefore, any inefficiencies in truck dispatching can have a cascading effect on the broader logistics network, incurring considerable delays and increased operational costs [3]. Secondly, truck dispatching in container ports is inherently a complex problem due to various constraints, including but not limited to time windows, truck capacities, and multiple loading and unloading locations within the port [10]. Traditional methods like heuristics and rule-based systems have been employed to address this problem; however, they often fall short in optimizing dispatching routes and schedules, particularly in dynamically changing conditions [5].

The utilization of GP for truck dispatching has emerged as a significant innovation, particularly in light of its ability to tackle intricate optimization tasks [8]. A variety of strategies have been developed to enhance GP performance, including specialized approaches such as double-layer GP [7], neural-network-assisted GP [9], cooperative coevolutionary GP [14], and reinforcement learning assisted GP [20]. These methodologies have collectively demonstrated commendable performance on comparable optimization problems. Despite these advancements, a salient limitation across the spectrum of GP-based methods is their computational demand, primarily owing to the necessity for comprehensive simulations to evaluate the fitness of individuals within the GP population [11].

Consequently, the integration of GP and surrogate models have been explored in previous research [15, 22, 21]. These studies capitalize on the computational efficiency of surrogate models to approximate fitness functions, thereby streamlining the optimization process [1]. Although these surrogate methods commonly employ simplistic mathematical models that suffice for relatively straightforward problems, they falter when applied to complex tasks such as container port truck dispatching. Empirical evidence from our experiments indicates that surrogate approaches based on simplistic mathematical models are ineffective in accurately approximating the complex dynamics of the port vehicle assignment problem. This inadequacy tends to obfuscate the evolutionary process in GP, complicating its convergence and subsequently diminishing the algorithm's overall performance.

Based on empirical findings, we contend that a more nuanced modeling technique is warranted for constructing a surrogate model capable of adequately simulating the complexities inherent in dry-port dispatching scenarios. The transformative impact of the transformer model [16] across various domains—including but not limited to natural language processing [18], image processing [4], and data prediction [17]—underscores its robust data-fitting and handling capabilities. Consequently, we propose the transformer model as the most suitable candidate for developing a surrogate model tailored to the specific challenges of container port truck dispatching.

Therefore, the present paper focuses on the innovative utilization of the transformer model to address the complex task of truck dispatching in container ports. Specifically, we introduce a surrogate model as an acceleration mechanism for the evolutionary process in Genetic Programming, culminating in a TSGP framework. This paper endeavors to fill existing gaps in the literature by integrating the distinct advantages of both GP and transformer models to achieve enhanced operational efficiency in the dynamic truck dispatching context at container ports.

3 Methodology

This section begins by detailing the complex dynamics and objectives inherent in the problem of truck dispatching within container ports. Building upon existing methodologies, which primarily employ manual heuristics and GP for formulating dynamic dispatching algorithms, we offer a nuanced discussion of these conventional approaches. To advance beyond the limitations of current frameworks, we introduce an innovative transformer-surrogate model specifically engineered to simulate truck dispatching processes within container port environments. This model is then integrated with GP, resulting in the formulation of the TSGP approach. This groundbreaking methodology aims to synergistically leverage the strengths of both GP and transformer models to tackle the multifaceted challenges embedded in this complex optimization problem.

3.1 Problem Description and Objective Function

This study focuses on optimizing dynamic truck dispatching in container ports to minimize ship docking time. Specifically, we address how allocating trucks to QCs and YCs can streamline container loading and unloading. Container sizes are standardized to twenty-foot equivalent units (TEUs), with trucks carrying a maximum of 2 TEUs. Dispatch tasks are bifurcated into single large container and dual small container assignments, with the latter requiring double the loading and unloading time.

QCs can handle either two smaller or one larger container per cycle, while YCs can only manage one. The sequential processing of containers is mandated by load-balancing needs and specific stacking location constraints. Typically, truck congestion and delays arise from front-positioned trucks dominating crane activities or lagging preceding tasks.

Given the port's strict routing regulations and the need for precise calculations related to queuing and waiting times, the internal truck dispatching challenge can be framed as a multi-constrained vehicle routing problem. It involves intricate computations within a directed graph model, primarily centered around minimizing idle times and expediting container throughput.

Let n designate the total number of dispatch tasks, $size_i$ signify the size of task i, E represent the ensemble of task end times, and S indicate the set of task start times. The objective function governing this dynamic truck dispatching problem can be formally expressed as:

$$\max\left(\frac{\sum_{i=1}^{n} size_i}{\max E - \min S}\right) \tag{1}$$

The objective function aims to maximize the task completion rate, measured in TEUs per hour, while minimizing total task duration. Due to its NP-hard complexity, as detailed in our prior work [7], straightforward mathematical modeling is unsuitable. To tackle this, we employ a dynamic dispatching framework that efficiently prioritizes tasks for idle trucks, thereby enhancing overall operational efficiency for both QCs and YCs.

3.2 Manual Heuristics

In most ports, dynamic truck scheduling predominantly relies on a hybrid system that integrates human oversight with carefully crafted manual heuristics shown in Algorithm 1, as discussed in a previous study [5]. This method, serving as the baseline in this paper, has shown effectiveness in contexts like Ningbo Port. However, it is limited by its simplistic architecture and narrow parameter set, often requiring domain-specific expertise for adjustments to evolving operational landscapes. We propose the GP-based dynamic dispatching method to address these limitations and serve as a point of comparison.

Algorithm 1 Manual Heuristic Truck Dispatching Algorithm

```
Require: Parameters $parameter$, Travel Time $t$
  function $heuristic(QC, truck)$
    if $crane\_truck\_num < desired\_trucks$ then
      $score \leftarrow travel\_time * (truck\_num - prority)$
    else
      $score \leftarrow travel\_time * desired\_trucks$
    end if
    if $truck\_num \geq truck\_limit$ then
      $score \leftarrow score + 200000$
    end if
    return $score$
  end function
```

3.3 Genetic Programming

GP is a specialized machine learning technique inspired by biological evolution employed in this study to optimize computer programs for specific tasks. It uses a predefined fitness function to evaluate a program's efficacy, following the principle of iterative improvement by continuously updating a population of solution candidates [12]. We have chosen the tree structure representation within the GP framework for this study. This choice is based on its intuitive nature and proven efficacy in port dispatch simulations, where the fitness of each solution is assessed through a specialized objective function as defined in Equation (1). Evolutionary mechanisms like mutation and replication refine these solutions, eventually converging on the one with the highest fitness score as the optimal output.

Building on our previous work [7], this study adopts the categorized Logical Genetic Programming (LGP) as the GP framework. LGP employs an extended set of operators, including arithmetic, relational, and logical functions, to offer enhanced performance and flexibility. These configurations are consistent with our previous research, where LGP demonstrated superior results.

In the specific domain of dynamic truck dispatching, GP is used to formulate robust and efficient heuristics. Like manual heuristics, environmental parameters are incorporated into the GP model to rank and optimize tasks. However, conventional GP often suffers from limitations, such as insufficient evolutionary generations within a fixed time frame, leading to poor performance and lack of convergence. To address this, we introduce a transformer surrogate model as an alternative to traditional simulation-based fitness evaluation, aiming for more efficient and effective algorithmic training.

3.4 Transformer Surrogate Model

The limitations inherent in conventional GP implementations, specifically regarding the restricted number of evolutionary generations in unit time and frequent converging failures, necessitate a more efficient approach to evaluating individual fitness. To address this challenge, we introduce a transformer surrogate model as a viable alternative to the traditional simulation-based fitness evaluator.

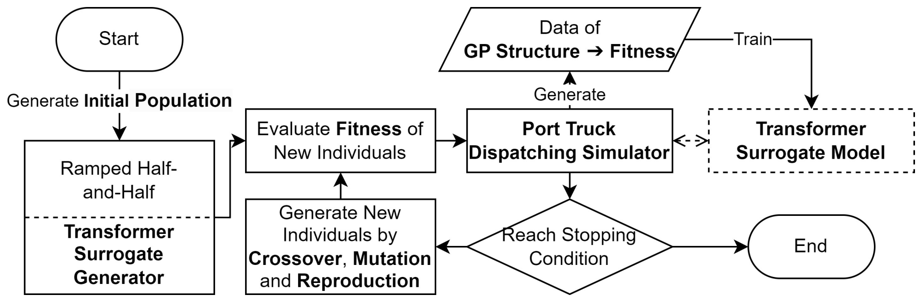
Start
Generate Initial Population
Ramped Half-and-Half
Transformer Surrogate Generator
Evaluate Fitness of New Individuals
Generate New Individuals by Crossover, Mutation and Reproduction
Data of GP Structure ➔ Fitness
Train
Generate
Port Truck Dispatching Simulator
Transformer Surrogate Model
Reach Stopping Condition
End

Fig. 2. GP Evolving Process

A transformer model, initially proposed for natural language processing tasks [16], has demonstrated its versatility and efficacy across various domains. In this study, the transformer model is a surrogate for the conventional simulation, providing rapid and accurate evaluations of GP individual candidates. As shown in Fig. 2, in the process of GP evolving, all newly generated individuals have to go through the port truck dispatching simulator to compute their fitness. Even though this part of the work can be calculated in parallel using multi-threaded computation, it still consumes many computational resources. Therefore, we propose to use a transformer surrogate model instead of the port truck dispatching simulator to accelerate the GP training. The transformer model in the paper follows the default model of the Transformers [19] except for changing the expected and actual output to one double float. According to our experimental results, using the transformer surrogate model can reduce the fitness computation time by more than 65% and is not affected by the increase in the number of tasks.

For the training of the transformer surrogate model, we propose to use the fitness data corresponding to the GP individual structure generated during the GP evolution process to train the transformer directly so that it does not need to consume time to produce data to prepare the transformer. The training process of the transformer can be carried out simultaneously with the GP evolution process, so in this paper, training the transformer surrogate model does not increase the training time of the algorithm. The transformer training process can be carried out simultaneously with the GP evolution process, so in this paper, training the transformer surrogate model does not increase the training time of the algorithm.

In addition, for the trained transformer surrogate model to perform well on datasets with different operating environments and to eliminate the interference of operating environments, the transformer surrogate model will not directly output the same TEU/h data as the simulator but will instead generate a number between 0 and 1, which represents the corresponding size of the GP individual's fitness. Once trained, the model estimates the fitness of new GP individuals with a substantially reduced computational cost.

3.5 Transformer-Surrogate Genetic Programming

After constructing the transformer surrogate model, we integrate it with GP to formulate the TSGP algorithm. As delineated in Algorithm 2, the evolutionary process of TSGP is carried out in multiple stages. Initially, up to generation x, fitness values for GP individuals are computed using a traditional simulator, while simultaneously training the transformer surrogate model with the newly generated data. Subsequently, for y generations, the fitness computation shifts to the transformer surrogate model. After this, the algorithm reverts to the simulator for z generations, updating the surrogate model in tandem. This loop continues unless the termination criteria are met.

The hyperparameters x, y, and z can be tuned according to the problem's inherent complexity and optimization objectives. This study empirically sets these parameters to 100, 100, and 10, respectively.

Building upon TSGP, we extend the algorithm to TSGP*, an enhanced version that leverages the trained transformer surrogate model for generating a more effective initial population. Unlike the traditional ramped half-and-half initialization method in standard GP, TSGP* employs a transformer surrogate generator. As elaborated in Algorithm 3, this generator also uses the ramped half-and-half technique to produce initial GP individuals. However, following their generation, the trained transformer surrogate model is invoked to estimate their fitness values. Only those individuals surpassing a fitness threshold f are retained. The value of f is a tunable hyperparameter set to 0.5 in this study.

In the subsequent section, we shall empirically assess the efficacy of both TSGP and TSGP* algorithms by applying them to a real-world case study in-

Algorithm 2 Transformer-Surrogate Genetic Programming Evolving

```
1: Initialize: GP Population, Transformer Surrogate Model
2: $t \leftarrow 0$                                        ▷ Initialize training generation counter
3: while Training time or number of generations not reached do
4:     if $t < x$ then                                     ▷ First $x$ generations
5:         Compute Fitness of GP individuals using Simulator
6:         Update Transformer Surrogate Model with Individual - Fitness Data
7:     else if $t \mod (x + y + z) < x + y$ then           ▷ Next $y$ generations
8:         Compute Fitness of GP individuals using Transformer Surrogate Model
9:     else                                                ▷ Next $z$ generations
10:        Compute Fitness of GP individuals using Simulator
11:        Update Transformer Surrogate Model with Individual - Fitness Data
12:    end if
13:    Perform GP operations (Selection, Crossover, Mutation)
14:    $t \leftarrow t + 1$                                ▷ Increment generation counter
15: end while
```

Algorithm 3 Transformer Surrogate Generator

```
1: Initialize: Empty GP Population, Transformer Surrogate Model
2: $m_{\text{count}} \leftarrow 0$                         ▷ Initialize individual counter
3: $f \leftarrow$ Fitness threshold
4: while $m_{\text{count}} < m$ do                         ▷ Until $m$ individuals are generated
5:     individual $\leftarrow$ Generate GP tree using ramped half-and-half
6:     fitness $\leftarrow$ Evaluate fitness using Transformer Surrogate Model
7:     if fitness $> f$ then
8:         Add individual to GP Population
9:         $m_{\text{count}} \leftarrow m_{\text{count}} + 1$
10:    end if
11: end while
```

volving dynamic truck dispatching in container ports. This performance evaluation will juxtapose multiple state-of-the-art methodologies, facilitating a comprehensive comparative analysis.

4 Experiment and Discussion

This section is devoted to a comprehensive evaluation of a diverse array of algorithms, specifically Data-Driven Genetic Programming (GP) [8], Deep Reinforcement Learning Hyper-Heuristic (DRL-HH) [23], Neural Network Assisted Genetic Programming (NN-GP) [9], TSGP, and TSGP*. The primary focus rests on elucidating the performance merits of TSGP and TSGP*, emphasizing the collaborative efficacy of amalgamating GP and transformer surrogate models. Given its robust and empirically validated performance, the manual heuristic [5] is the chosen benchmark for this comparative study.

The feature set used in this study, often termed GP terminals, closely follows the specifications outlined in our prior research [7]. It encompasses 14 distinctive features, such as truck travel time, the current quantity of QC trucks, the number of trucks in waiting, and the remaining task count, among other salient variables, to accurately portray the operational state of the container port at any given moment. The GP algorithm is configured with a population size of 1024 and employs crossover, mutation, and reproduction rates of 60%, 30%, and 10%, respectively. All the algorithms in this study were subjected to a *10-hour training regimen.* . The datasets leveraged in this experiment are consistent with those used in our previous work [7], originating from historical operations at Ningbo-Zhoushan Port. The experimental setup mimics a real-world single ship berth featuring six QCs, with the number of trucks being variable and reflecting the actual operational conditions, ranging between 24 and 48. Uncertain variables like truck travel time and container loading and unloading times are modeled based on empirical distributions derived from genuine operational data.

This study categorized historical task records into ten sets, each representing a distinct operational scenario encountered at different temporal intervals. Five sets were used for training, while the remainder were for testing. Each training or testing set consists of ten instances from a similar time frame, incorporating 200 tasks that include a blend of loading and unloading operations. Each training instance was executed 100 times, employing different random seeds for each run. A comprehensive summary of the average training and testing results is provided in Table 1.

In light of varying benchmark performances observed in real-world data, we introduce the Improvement over the Manual Heuristic (Imp.) as a reference baseline. All subsequent performance metrics are framed as improvements over this established baseline. A t-test conducted on the experimental outcomes ($\alpha = 0.001, p = 0.00$) substantiates that the fully realized TSGP and TSGP* variant exhibits notable performance enhancements over GP, DRL-HH, and NN-GP counterparts—registering an improvement of approximately 7%, 5%, and 3% on the training and test sets, respectively. Notably, the TSGP model demonstrates

Table 1. GP, DRL-HH, NN-GP, TSGP and TSGP* Experiment Results in 10 Training Hours (TEU/h)

Set	No.	Manual	GP	DRL-HH	NN-GP	TSGP	TSGP*
Train	1	106.87	118.76	117.17	117.84	122.05	130.84
	2	126.85	133.84	139.85	145.21	147.99	149.01
	3	114.27	120.76	127.00	126.14	132.30	134.36
	4	106.31	113.78	112.91	117.14	123.55	122.31
	5	116.88	129.17	125.75	132.60	133.90	136.91
	Avg.	114.236	123.26	124.53	127.79	131.96	134.68
	Imp.	**0.00%**	**7.90%**	**9.02%**	**11.86%**	**15.51%**	**17.90%**
Test	1	105.87	113.05	115.01	111.19	118.70	119.46
	2	118.91	125.93	126.59	134.01	135.50	133.47
	3	115.72	122.58	124.48	126.06	133.81	133.78
	4	121.48	130.10	132.59	134.92	135.93	139.37
	5	117.25	120.63	125.28	127.15	135.72	132.23
	Avg.	115.846	122.46	124.79	126.67	131.93	131.66
	Imp.	**0.00%**	**5.71%**	**7.73%**	**9.34%**	**13.89%**	**13.65%**

negligible performance degradation when transitioning from training to test sets. This underscores TSGP's robustness and effective generalization capabilities, rendering it highly applicable to previously unseen datasets.

To probe further into TSGP's potential, we also evaluated TSGP*, an alternative algorithmic formulation incorporating a transformer-based surrogate generator. Empirical evidence attests to substantial performance gains for TSGP* on both training and test sets. This highlights the efficacy of employing a transformer surrogate model in the initial population generation phase, enhancing the overall quality of the GP-based solutions. However, TSGP* does reveal a drawback—increased performance degradation in the test phase, possibly attributable to the lack of diversity in the initial population. Our future work will address this shortcoming to further optimize TSGP*'s efficacy across training and test environments.

In this comprehensive study, we have undertaken a dual-faceted evaluation approach to analyze the efficacy of various algorithms. The first facet involved a time-based comparison where each algorithm was subjected to a fixed training period of 10 hours. The second, more granular facet, involved an extended analysis over 300 evolutionary generations for each algorithm, as detailed in Table 2. This two-pronged approach was designed to provide a more holistic understanding of each algorithm's capabilities and limitations.

Our findings, as elucidated by the experimental data, indicate a nuanced spectrum of performance across the algorithms tested. TSGP, when assessed over a congruent number of evolutionary generations, exhibited performance that did not meet the optimal benchmarks. This was largely due to the variation between the real-world operational conditions and those within the simulated environment—a factor that often plagues algorithmic simulations.

However, an intriguing pattern was observed with TSGP*, which integrates the Transformer architecture for initial population generation. This integration appears to confer an advantage, allowing TSGP* to withstand the discrepancies between simulation and reality, thus not significantly affecting its performance.

Moreover, an examination of the time metrics reveals a compelling insight. While heuristic and GP-based methods demonstrate only marginal variations in time consumption across the test set, the training efficiency of both TSGP and TSGP* emerges as distinctly superior. This efficiency is not merely a matter of reduced time expenditure; it is indicative of the depth and adaptability of the algorithms themselves.

The results highlight the proposed surrogate method's profound impact, particularly its ability to accelerate the training process significantly. It is a compelling argument for the surrogate method's integration into the evolutionary computation framework. The method streamlines the computational workflow and ensures that the algorithms remain robust in the face of environmental variability. This is particularly relevant in real-world applications where such variability can often be unpredictable and challenging to simulate.

Table 2. GP, DRL-HH, NN-GP, TSGP and TSGP* Experiment Results in 300 Training Generations (TEU/h)

Set	No.	Manual	GP	DRL-HH	NN-GP	TSGP	TSGP*
Train	1	106.87	115.2474	118.2288	128.6010	120.9443	128.0169
	2	126.85	132.9704	132.4145	135.4286	132.0230	138.4152
	3	114.27	121.0681	132.0767	127.8843	121.7983	133.1774
	4	106.31	118.7571	119.2456	125.5672	117.5625	125.2360
	5	116.88	127.9977	122.5157	132.5091	128.0955	129.9707
	Avg.	114.236	123.2081	124.8962	129.9980	124.0847	130.9632
	Imp.	**0.00%**	**7.85%**	**9.33%**	**13.80%**	**8.62%**	**14.64%**
	Time(hour)	0.00	7.71	22.21	13.58	5.63	5.85
Test	1	105.87	115.8348	118.4098	118.7419	112.3532	124.4371
	2	118.91	127.3668	128.1347	132.6995	126.7159	130.8406
	3	115.72	116.4109	122.1811	125.3569	120.2386	127.3589
	4	121.48	127.4641	125.8459	137.2236	127.5141	132.1725
	5	117.25	125.4904	125.8270	131.5146	129.7641	128.6010
	Avg.	115.846	122.5134	124.0797	129.1073	123.3172	128.6820
	Imp.	**0.00%**	**5.76%**	**7.11%**	**11.45%**	**6.45%**	**11.08%**
	Time(min)	0.31	0.35	0.49	0.51	0.42	0.46

In conclusion, the TSGP framework adeptly amalgamates the strengths of both the transformer and GP paradigms, showcasing superior performance in varied training and testing conditions. The model addresses the high simulation cost inherent in dynamic truck dispatching across multi-scenario ports, even amidst uncertainties. Given its successful implementation, TSGP holds substantial promise for broader applications in dynamic transportation optimization problems, opening up a wide avenue for future research endeavors.

5 Conclusion

This study introduces the TSGP approach, a groundbreaking methodology that addresses the computational inefficiencies in traditional GP applied to truck dispatching in container ports. By employing a transformer model as a surrogate evaluator during the fitness calculation stage, TSGP not only drastically reduces the algorithmic training time but also enhances the performance of GP. The transformer model further serves a dual role by generating optimized initial populations for GPs, demonstrating a synergistic integration of the computational strengths of transformer models with the heuristic search capabilities of GPs.

Our empirical results confirm the efficacy of TSGP, especially its ability to significantly accelerate the training process while maintaining or improving operational efficiency. The transformer model effectively learns to approximate the fitness landscape of the GP, thereby streamlining the evolutionary process. However, while TSGP and TSGP* showed robust performance in both training and testing scenarios, some challenges related to initial population diversity in our extended model, TSGP*, indicate directions for future research.

In summary, the TSGP approach marks a substantial advance in container port optimization, particularly in truck dispatching systems. By marrying the capabilities of transformer models and GPs, we open new avenues for applying machine learning techniques in operational logistics and beyond. The demonstrated success of TSGP and TSGP* sets the stage for their broader application across various domains where computational efficiency and effective heuristic search are critical.

References

1. Alizadeh, R., Allen, J.K., Mistree, F.: Managing computational complexity using surrogate models: a critical review. Research in Engineering Design **31**, 275–298 (2020)
2. Bank, W.: The container port performance index 2020: A comparable assessment of container port performance. World Bank (2021)
3. Bartolacci, M.R., LeBlanc, L.J., Kayikci, Y., Grossman, T.A.: Optimization modeling for logistics: options and implementations. Journal of Business Logistics **33**(2), 118–127 (2012)
4. Chen, H., Wang, Y., Guo, T., Xu, C., Deng, Y., Liu, Z., Ma, S., Xu, C., Xu, C., Gao, W.: Pre-trained image processing transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12299–12310 (2021)
5. Chen, J., Bai, R., Dong, H., Qu, R., Kendall, G.: A dynamic truck dispatching problem in marine container terminal. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8. IEEE (2016)

6. Chen, X., Bai, R., Dong, H.: A multi-layer gp hyper-heuristic for real-time truck dispatching at a marine container terminal. MISTA 2019 (2019)
7. Chen, X., Bai, R., Qu, R., Dong, H.: Cooperative double-layer genetic programming hyper-heuristic for online container terminal truck dispatching. IEEE Transactions on Evolutionary Computation (2022)
8. Chen, X., Bai, R., Qu, R., Dong, H., Chen, J.: A data-driven genetic programming heuristic for real-world dynamic seaport container terminal truck dispatching. In: 2020 IEEE Congress on Evolutionary Computation (CEC). pp. 1–8. IEEE (2020)
9. Chen, X., Feiyang, B., Qu, R., Jing, D., Bai, R.: Neural network assisted genetic programming in dynamic container port truck dispatching. In: 2023 IEEE International Conference on Intelligent Transportation Systems (ITSC). pp. 1–6. IEEE (2023)
10. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Management science **6**(1), 80–91 (1959)
11. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evolutionary computation **23**(3), 343–367 (2015)
12. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and computing **4**, 87–112 (1994)
13. Macharis, C., Caris, A., Jourquin, B., Pekin, E.: A decision support framework for intermodal transport policy. European Transport Research Review **3**, 167–178 (2011)
14. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming. IEEE Transactions on Evolutionary Computation **18**(2), 193–208 (2013)
15. Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. IEEE transactions on cybernetics **47**(9), 2951–2965 (2016)
16. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
17. Wang, T., Chen, J., Lü, J., Liu, K., Zhu, A., Snoussi, H., Zhang, B.: Synchronous spatiotemporal graph transformer: A new framework for traffic data prediction. IEEE Transactions on Neural Networks and Learning Systems (2022)
18. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations. pp. 38–45 (2020)
19. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45. Association for Computational Linguistics, Online (Oct 2020), `https://www.aclweb.org/anthology/2020.emnlp-demos.6`
20. Yi, W., Qu, R., Jiao, L., Niu, B.: Automated design of metaheuristics using reinforcement learning within a novel general search framework. IEEE Transactions on Evolutionary Computation (2022)
21. Zeiträg, Y., Figueira, J.R., Horta, N., Neves, R.: Surrogate-assisted automatic evolving of dispatching rules for multi-objective dynamic job shop scheduling using genetic programming. Expert Systems with Applications **209**, 118194 (2022)

22. Zhang, F., Mei, Y., Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. IEEE Transactions on Evolutionary Computation **25**(4), 651–665 (2021)
23. Zhang, Y., Bai, R., Qu, R., Tu, C., Jin, J.: A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. European Journal of Operational Research **300**(2), 418–427 (2022)
24. Zheng, F., Qiao, L., Liu, M.: An online model of berth and quay crane integrated allocation in container terminals. In: Combinatorial Optimization and Applications: 9th International Conference, COCOA 2015, Houston, TX, USA, December 18-20, 2015, Proceedings. pp. 721–730. Springer (2015)