Dynamic Landscape Analysis for Open-Ended Stacking

Bernhard Werth[1,2], Johannes Karder[1,2], Andreas Beham[1], Stefan Wagner[1]
[1]Josef Ressel Center for Adaptive Optimization in Dynamic Environments
University of Applied Sciences Upper Austria
Hagenberg, Austria
[2]Institute for Formal Models and Verification
Johannes Kepler University
Linz, Austria
bernhard.werth@fh-hagenberg.at

ABSTRACT

Fitness landscape analysis (FLA) is a useful tool in the domain of (meta-)heuristic optimization but depends on explicitly knowing what fitness value is assigned to each solution. Dynamic optimization problems often do not provide their fitness landscape in such an explicit form, but by employing problem-specific knowledge, information about the problem itself and its current state can still be obtained. In this paper, a type of gray-box analysis of states of the open-ended stacking problem in two variations is presented. The current states obtained by monitoring the problem and algorithm during optimization are described via statistical measures similar to FLA measures. From this the distribution of possible states (the state landscape) and the transitions between problem states are analyzed. Visualization of the empirically obtained results reveals insights into algorithm-problem dynamics.

CCS CONCEPTS

• **Theory of computation** → **Online algorithms**; *Random search heuristics*; • **Mathematics of computing** → *Combinatorial optimization.*

KEYWORDS

dynamic optimization, fitness landscape analysis, state space analysis

ACM Reference Format:
Bernhard Werth, Johannes Karder, Andreas Beham and Stefan Wagner. 2021. Dynamic Landscape Analysis for Open-Ended Stacking. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3449726.3463153

1 INTRODUCTION

Research in the evolutionary computation (EC) domain often yields methods/approaches that can be applied to both academic and real-world optimization problems. Scientific contributions can be found in various subareas of the research domain, ranging from classical static problem formulations to dynamic optimization problems. The latter are of importance in real-world optimization, since many real-world processes are dynamic by nature. For this very reason, dynamic optimization problems (DOPs) have become an increasingly important subject of research, e.g. in production and logistics optimization where scheduling and stacking problems are well researched, however, dynamic problems, where machine breakdowns can occur, tasks are not (fully) known beforehand and conditions like processing times and restrictions change, have not been explored as much [3, 10].

To gain more insights into the difficulty of problems or better understand the behavior of optimization algorithms, research has come up with the concept of *fitness landscape analysis*. Specific measures can be taken for various optimization problems, which help to categorize a set of problems and e.g. provide useful information for optimization algorithms or even find promising algorithm-problem pairs (algorithm selection) [1, 16]. However, to the best of our knowledge, FLA has only found very few applications in the domain of DOPs.

A major issue when applying FLA to DOPs is that the fitness of a potential solution at a fixed point in time cannot be measured directly, rather in many real-world problems only the performance of the whole optimization algorithm over longer periods can be assessed. While the fitness impact of a single decision can be gauged by restarting the problem and changing only this decision, this is often not feasible for real-world problems. Rather it would be prudent to combine domain-knowledge and information about the current state of the DOP with those FLA measures that can be estimated without requiring explicit fitness evaluations.

In this paper, we showcase the use of gray-box landscape analysis on an open-ended stacking problem. Here, gray-box means combining domain information (white-box) with "traditional" landscape measures where nothing except the evaluation function is known(black-box). Two academic problem formulations based on real-world applications are used to create a repeatable experiment. By calculating statistic measures describing the current problem state and grouping similar states together, we gain insights into the dynamics of the problem and the employed algorithm.

The rest of this paper is outlined as follows: Section 2 guides the reader towards existing literature concerned with the application of dynamic fitness landscape analysis. The dynamic optimization

GECCO '21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07.
https://doi.org/10.1145/3449726.3463153

problem is explained in Section 3. In Section 4, the analysis procedure is presented in detail. Conducted experiments and results are shown in Section 5. The paper concludes with Section 6.

2 LITERATURE REVIEW

For a certain subset of optimization problems, it is not enough to find a single solution by solving these problems once. Dynamic optimization problems (DOPs) change over time, which makes it necessary for optimization algorithms to not only find, but also track the optimum. Real-world examples of DOPs are dynamic vehicle routing [20, 22], dynamic scheduling problems [2, 18], dynamic container relocation problems [30] and dynamic machine configuration [12].

Several classifications of DOPs have been proposed [4, 6, 15]. In [15], the following key properties of DOPs are defined:

- Time-linkage: Whether the solution found by the optimizer has an influence on how the problem changes.
- Visibility: Whether the optimizer is notified about dynamic changes; for problems with different dynamic influences, hybrid visibility levels may exist.
- Periodicity: Whether the problem displays cyclic behavior and returns to previous states.
- Changing Factors: Objective functions, domain of variables, number of variables, constraints.

There exist a number of academic benchmarks for DOPs, such as the moving peaks problem [5], the dynamic XOR generator [29], the dynamic Knapsack problem [23] and dynamic NK-landscapes [8]. A generic problem generator for dynamic binary optimization problems which is specifically used for FLA was created by Tinós and Yang [25].

Most of the academic benchmarks focus on a changing objective function, while keeping most other factors, e.g. number and domain of variables, constant. Similar to all examined DOPs in the survey of [15], most academic problem formulations do not display time-linkage behavior. Conversely, several of the real-world DOPs listed above, especially the scheduling problems, exhibit time-linkage and changing solution encodings, i.e. number and domain of decision variables also vary over time.

Fitness landscape analysis (FLA) aims to capture significant characteristics of an optimization problem [21]. These may provide insights into the problem itself [17], the behavior of the optimization algorithm [27] and enable the automated selection of appropriate algorithms [1]. Well known static FLA measures include:

- Ruggedness measures, including auto-correlation [28], are often calculated using *walks* through the fitness landscape.
- In [14], a set of measures trying to describe problem hardness is reviewed.
- Epistasis [9] describes how strongly decision variables are connected and is often analyzed in context with NK landscapes [11].
- Local optima networks capture the transitions between locally optimal solutions [17] and several measures can be calculated using the resulting graph.

Compared to static optimization problems, only relatively few fitness landscape measures have been proposed for dynamic optimization problems. These include *change severity*, *mean optimal fitness difference*, *(mean) fitness distance correlation* [7, 25] and *rank-based difference* [24], which compare solution qualities and the location of the global optima before and after a discrete dynamic event. In [13], *stationarity of amplitude change*, *keenness*, *periodicity* and *change degree of average fitness* are calculated for static landscapes, and the mean difference of these measures before and after events is used to describe the dynamicity of the problem. Furthermore, the *fitness landscape similarity* after dynamic time warping is proposed.

3 OPEN-ENDED STACKING

In the spirit of reproducibility, the stacking problems from the 2021 GECCO Competition "Dynamic Stacking Optimization in Uncertain Environments"[1] are used as a benchmark for this paper.

3.1 Hot Storage Scenario

Figure 1 (upper) shows a possible state of a hot storage (HS) warehouse in steel production during optimization/simulation. The depicted warehouse consists of one arrival stack on the left, several buffer stacks in the middle and one handover area on the right. New blocks spawn at unknown times at the bottom of the arrival stack, shifting all existing blocks there upwards until a maximum stack height is reached. If the arrival stack is full, the arrival process blocks until blocks are removed – this is undesired, as in the real world preceding production processes might be blocked as well. The *blocked arrival time* is one the five objectives of the optimization problem and should be kept minimal while the number of *delivered blocks* should be maximized. Additionally, the operation of the crane incurs its own cost, so the number of *crane manipulations* (moves) should be kept minimal.

All buffer stacks are also limited in height. Once reached, no blocks can be dropped on the respective stack anymore. Each block carries its own due date, at which it must be handed over. *Overdue blocks* are undesired and constitute the fourth objective. For a successful handover, both block and handover stack must be in a ready state. It is unknown when a block or the handover stack transitions into the ready state, however, it is guaranteed that a block will become ready before its due date is reached. The handover stack becomes unavailable after a successful handover and resets into the ready state after some time, limiting the rate at which blocks can leave the system. Only one block can be handed over at a time and the crane can only carry one block at a time. Once a block has been put on the handover, it cannot be removed by the crane anymore.

As an additional source of uncertainty and dynamicity, the time the crane takes to perform any action is stochastic. The crane cannot be interrupted during a move, it can only be interrupted between moves, limiting the times at which the optimizer can effectively react to dynamic changes.

From an optimizer perspective, the problem presents itself as a stream of world states that have to be answered with a stream of crane instructions. Crane instructions take the form of lists of moves $(source, target, blockId)$ that the crane processes sequentially. The crane can be relocated without moving a block by using empty moves (moves without a $blockId$). If the crane has currently no instructions, it will idle. This makes the time at which the optimizer

[1] https://dynstack.adaptop.at/

Figure 1: The hot storage scenario with a single crane (upper) and the rolling mill scenario with two cranes (lower).

publishes its schedule a decision variable in itself. Sending a new set of crane instructions overrides the current schedule, except for the move that is currently carried out.

3.2 Rolling Mill Scenario

The rolling mill scenario (RM) is depicted in Figure 1 (lower) and differs from the hot storage in a few aspects. The warehouse has multiple arrival stacks and handover locations. A handover location belongs to a rolling mill where blocks exiting the warehouse are processed further. Each rolling mill has a milling program that dictates which blocks should be delivered and milled next. Multiple blocks can be dropped off at a handover location. Blocks dropped in the wrong sequence or on the wrong handover location will be processed by the rolling mill, but counted as *mess-ups*. The number of mess-ups is the last objective should be kept minimal (The number of mess-ups for the hot storage scenario is inherently 0). Therefore, it is imperative to deliver the blocks to their correct handover locations in order. The respective milling program of each block is known. However, the point in time when a block will be milled is only known minutes before the actual milling process is started.

Furthermore, in this warehouse two cranes are operated (cf. C1 and C2 in Figure 1). They share a single crane lane, thus making it necessary to coordinate crane movements as it is impossible for one crane to overtake the other. No crane can reach all stacks within the warehouse alone, necessitating transshipments. Both cranes can access all buffer stacks, however, only the shuffle crane (C1) can access the arrival stack and only the handover crane (C2) can access the handover stacks.

Finally, compared to the hot storage scenario, the cranes in the rolling mill scenario are able to carry more than one block at a time.

3.3 Problem Characteristics

From a dynamic optimization point of view, the dynamic stacking problems display different characteristics compared to academic benchmark functions, such as the moving peaks problem. As most benchmarks change their fitness landscapes after a certain number of function evaluations, it is possible to evaluate multiple solutions for the same fitness landscape. The dynamic optimization problem presented here requires the optimization process to decide for itself which potential schedules should be favored over others using only domain specific knowledge. Furthermore, the dynamic scheduling problems are time-linked, meaning that the optimization results themselves change the fitness landscape. Lastly, the qualities of a single solution computed at different points in time often cannot be easily compared, because, as the problem changes, solutions quickly become invalid (i.e. the solution encoding changes over time). Thus several FLA measures such as *fitness correlation* that depend on comparing qualities before and after dynamic changes cannot be calculated. Similarly, walk-based or sampling-based FLA measures are difficult to evaluate because of the aforementioned time-linkage. Finally, assessing the quality of a solution is not an instant action and punishing or beneficial effects of made moves are reflected in the cumulative quality indirectly and much later when the fitness landscape has long changed.

4 STATE LANDSCAPE ANALYSIS

To explore the relevant states of the DOP, a straight forward way is to apply an optimizer to the problem and record the problem's state at defined points in time, e.g. every n seconds or whenever event e happens. For the following results, the problem state was queried every second for one hour of simulated time. In order to make problem states comparable at different points in time or between different optimizer runs, states need to be described via their statistical properties. Table 1 shows a list of features comprising the state descriptors. Roughly speaking, these features can be grouped into:

- features that are only dependent on the warehouse geometry and therefore *constant*
- features that describe the *current* state of the warehouse
- features that represent the optimization *objectives* at the current time
- features describing how the state has changed in the last n seconds (*dynamic*)
- features that take the view of the optimizer (i.e. the fitness landscape) into account

Applying static and dynamic fitness landscape measures to the open-ended stacking problem is not trivial, as most measures require an explicit mapping between solutions and their associated performances. However, in the open-ended stacking problem, only the cumulative performance of the solver over time is explicitly known.

However, the prominent dynamic fitness landscape measure named *change severity* and *change frequency* can be approximated. Change severity, as defined in [7], is the difference between the optimal solutions before and after a dynamic change event, which is normalized by the maximum achievable distance in the search space. In the same paper, the difficulty of finding the global optimum in real-world problems is acknowledged and the change severity is approximated by comparing the best solutions of an evolutionary algorithm before and after the change event. Since the open-ended stacking problem undergoes continuous change, rather than discrete change events, change severity must be defined per-time unit, e.g. on a per second basis. Furthermore, although the quality of the optimal solution is unknown, the solution the optimizer deems to be optimal is known, which can be substituted as the location of the global optimum.

For any given current problem state, a solution is defined as a set of n next moves that are to be executed by the specified crane(s). Following the problem reduction for *static* crane scheduling problems in [19], we consider only the sequence of moves as relevant, not their specific starting times. Therefore, the change severity for the open-ended stacking problem can be approximated by

$$severity_n = \frac{d(\boldsymbol{x}_{n,t}, \boldsymbol{x}_{n,t+\delta})}{\delta} \tag{1}$$

where $\boldsymbol{x}_{n,t}$ are the next n moves emitted by the optimizer at time t and δ is the time horizon over which the change severity is measured. For optimizers that provide fewer than n next moves, empty moves are imputed. For simplicity, d was chosen to be the hamming distance. In the following experiments, the change severity was measured at 1 Hz and the mean change severity over the last 2 minutes was computed ($n = 1, 5, 10$). These three change severity measures are used in conjunction with the other (domain-dependent) measures to create statistical descriptors of the current problem state.

In order to visualize and describe the behavior of a given solver-problem pair, the transitions between problem states must be taken into consideration. Therefore, a dynamic state network is proposed. By assuming that throughout the optimization process, some encountered problem states are sufficiently similar, although not necessarily consecutive in time or not even occurring in the same optimization run, state descriptors can be sub-sampled via clustering. For this, it is important that all state features are time-invariant or normalized accordingly. By using the resulting clusters as nodes and the encountered transitions between clusters as edges, a characteristic, directed graph for further analysis can be created.

5 EXPERIMENTS AND RESULTS

To be able to extract the proposed landscape measures, both the hot storage and rolling mill setups were used with three different warehouse layouts (number of stacks, maximum stack height, ...). For each rolling mill layout three different dynamic parametrizations (distribution of arrivals, times until blocks are due, ...) and for the hot storage four parametrizations were tested. This yields a total of 21 experimental simulation setups used to produce landscape measures. On each experimental setup, a rule-based optimizer was allowed to operate for one hour of simulation time. Experiments were repeated 30 times. In order to achieve more stable landscape measures, measures were extracted every second and aggregated into (non-overlapping) bins of 30 seconds via averaging, yielding a *state descriptor* consisting of 37 measures for every half a minute

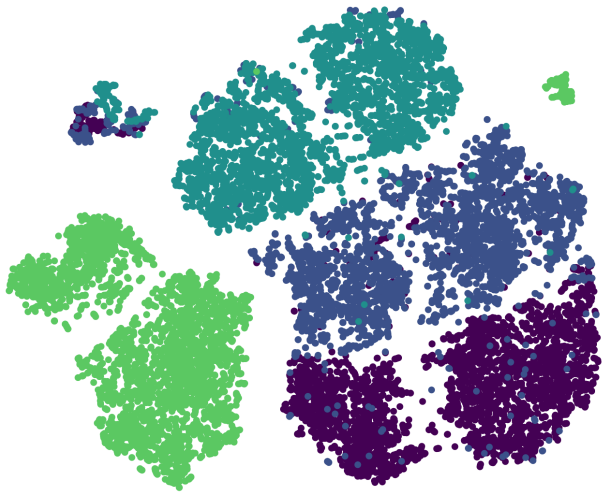

Figure 3: HS-3 Stacks colored by scenario parametrization

of simulation time. The five objectives themselves accumulate over time and are therefore not suitable for detecting similar (revisited) states.

5.1 Result 1: System State Similarities

Figure 2 shows t-SNE [26] projections of the obtained descriptors (using unweighted Euclidean distance) grouped by warehouse layout. Note that t-SNE is a non-parametric non-linear projection of the descriptor space that focuses on maintaining neighborhoods. The rotation and size of the image contain no information and points in one cluster may be closer to each other than in other clusters. For all layouts a few clearly separated dark clusters containing mostly descriptors from the very beginning of the optimization are shown. This indicates that the initial state of the warehouse from which the optimizer has to start from is *(substantially) different from the later operating states*, when the optimizer had a chance to find a stable working regime. All HS projections further display a smaller number of less separated, but still *identifiable clusters* with no apparent correlation to the amount of time passed. Figure 3 explains some of these clusters as descriptors pertaining to different dynamic parameter settings are almost disjoint. However, for HS-3 Stacks most parameter setting induced sections display two separated partitions which are neither explained by parameter setting nor by optimization time. Another prominent feature are the *"string"-like structures* appearing around the center cluster in all RM variations. These strings represent descriptors of problem states where the algorithm encountered *unstable configurations* (e.g. warehouse too full) where barely any moves are executed and the overall "tardiness" in the system increases. Therefore the state descriptors yield very similar measures, except for those measures that take tardiness into account and progress linearly with time.

5.2 Result 2: System State Transitions

The analysis of the state descriptors as individual points, can be enhanced when taking their transition function into account. In order to analyze which states lead to which, the descriptors need to be grouped (as descriptors are very rarely exactly the same). In this paper, the descriptors were clustered via hierarchical clustering (euclidean distance, ward-linkage and 200 fixed clusters) into nodes of a directed graph. The edge weight was set by counting how often descriptors of one cluster lead to descriptors of another cluster. Figure 4 displays these graphs for all warehouse layouts. The spring-layout used to plot these graphs positions nodes with stronger connections (more and heavier edge-weights) closer together. In contrast to the t-SNE projection which clusters descriptors according to their numeric similarity, this method groups them by their transitional probability. Reflexive transitions constitute the majority of transitions and are not plotted for visibility reasons.

For the HS scenario the graphs show two visibly *distinct components* of nodes that are highly connected within each other. As indicated by the colors, this shows that the operating states for one set of dynamic parameters are contained in a separate area of the state space, while the other three settings share a more connected though divisible area.

The RM graphs display similar shapes as their corresponding t-SNE plots with a larger well connected group of nodes in the center and scattered clusters around. No edges lead from these outer clusters back towards the center, indicating that at least this algorithm is not able to recover from a degraded state.

6 CONCLUSION

In this paper, a type of gray-box analysis for dynamic stacking optimization problems, for which no fitness landscape is explicit, is introduced and demonstrated on two different dynamic stacking/crane scheduling problems with multiple layouts and dynamic configurations. The proposed method contains multiple state measures including approximations of the established FLA measures *change severity* and *change frequency* which are calculated parallel to the execution of any optimization algorithm and do not require fitness evaluations or interfere with the search. Hence, performing this state analysis does not decrease the performance of the employed solver, assuming the overhead in execution time is negligible. Another major advantage is that state landscape analysis can be applied when fitness values are only obtained in a cumulative and implicit form and the quality of individual decisions is difficult to discern. Visual exploration of the traversed state space indicates that this type of analysis can provide insights into the dynamics of the problem and the algorithm, for example, how the search process revisits certain states or fails to recover from specific situations, which might indicate the need to switch to a different algorithm.

Future work will include:

- the extension of the state descriptors with more measures
- the prediction of future algorithmic performance
- the selection or weighting of state measures based on their variable impact in these prediction models in order to combat co-linearities and lessen the impact of multiple similar or less informative features
- the inclusion of proxy fitness-evaluation (e.g. static approximations of the dynamic problem)

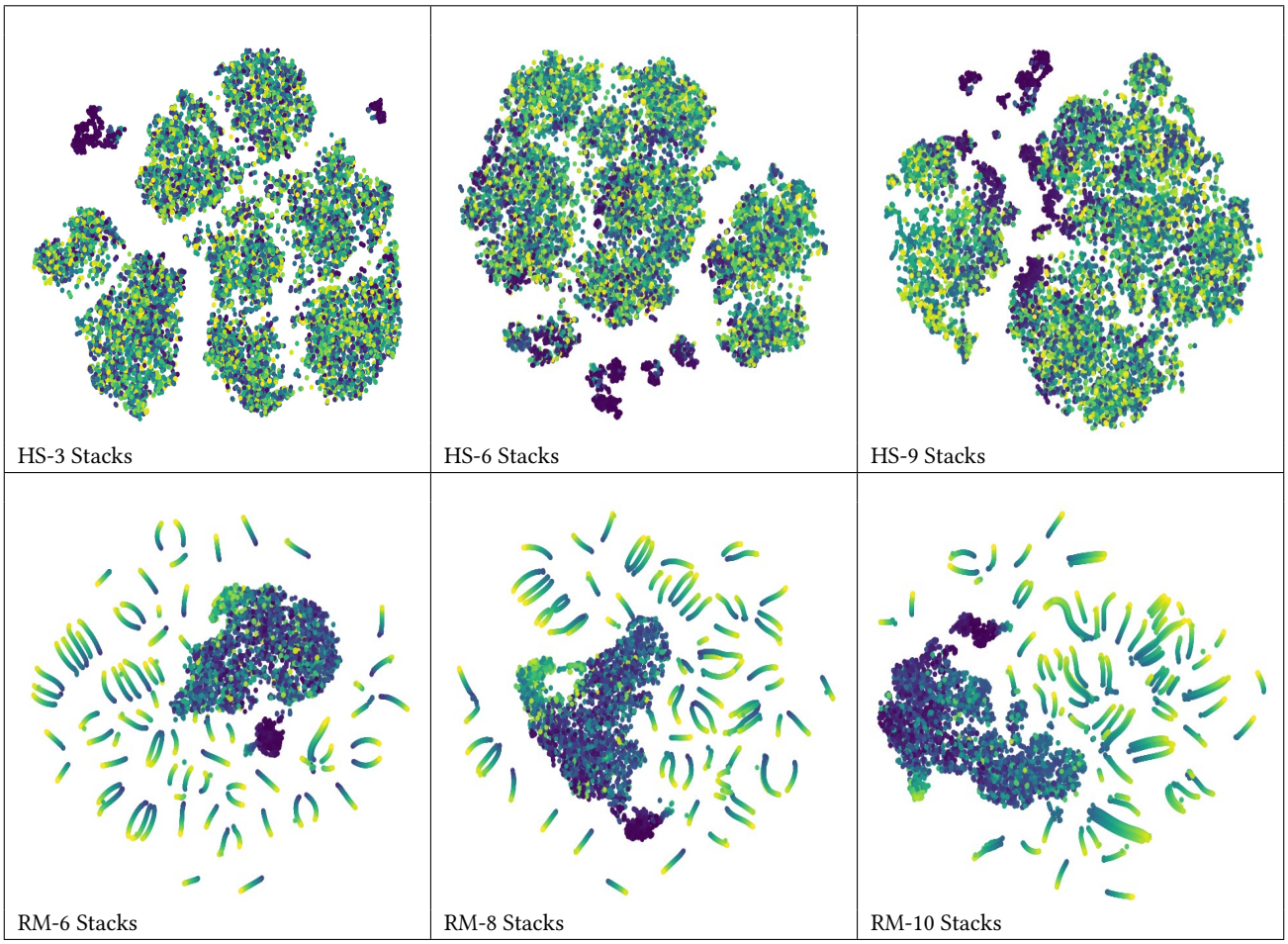

Figure 2: System State Similarities - t-SNE projections of the state descriptors grouped by warehouse geometries and colored by time. Dark colors indicate early states while light colors indicate later ones.

- the application to scenarios where the environmental circumstances (e.g. handover readiness, crane availability) undergo dynamic changes (concept drifts)
- the analysis of state spaces of different and more complex optimization algorithms

Finally, we believe open-ended dynamic optimization can benefit majorly from automated algorithm selection and parametrization and the information obtained from state space analysis can be used, similarly to how conventional fitness landscape analysis features are used in the static case.

ACKNOWLEDGMENTS

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs and the National Foundation for Research, Technology and Development is gratefully acknowledged.

REFERENCES

[1] Andreas Beham, Michael Affenzeller, and Stefan Wagner. 2017. Instance-based algorithm selection on quadratic assignment problem landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1471–1478.

[2] Jacek Błażewicz, Klaus H Ecker, Erwin Pesch, Günter Schmidt, Małgorzata Sterna, and Jan Weglarz. 2019. *Handbook on Scheduling: From Theory to Practice*. Springer.

[3] Nils Boysen, Dirk Briskorn, and Frank Meisel. 2017. A generalized classification scheme for crane scheduling with interference. *European Journal of Operational Research* 258, 1 (2017), 343–357.

[4] Jürgen Branke. 1999. *Evolutionary algorithms for dynamic optimization problems: A survey*. AIFB.

[5] Jürgen Branke. 1999. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, Vol. 3. IEEE, 1875–1882.

[6] Jürgen Branke. 2012. *Evolutionary optimization in dynamic environments*. Vol. 3. Springer Science & Business Media.

[7] Jürgen Branke, Erdem Salihoğlu, and Şima Uyar. 2005. Towards an analysis of dynamic environments. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. 1433–1440.

[8] Roger Eriksson and Björn Olsson. 2004. On the performance of evolutionary algorithms with life-time adaptation in dynamic fitness landscapes. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, Vol. 2. IEEE, 1293–1300.

[9] Cyril Fonlupt, Denis Robilliard, and Philippe Preux. 1998. A bit-wise epistasis measure for binary search spaces. In *International Conference on Parallel Problem Solving from Nature*. Springer, 47–56.

[10] Viktoria A Hauder, Andreas Beham, Stefan Wagner, Karl F Doerner, and Michael Affenzeller. 2021. Dynamic online optimization in the context of smart manufacturing: an overview. *Procedia Computer Science* 180 (2021), 988–995.

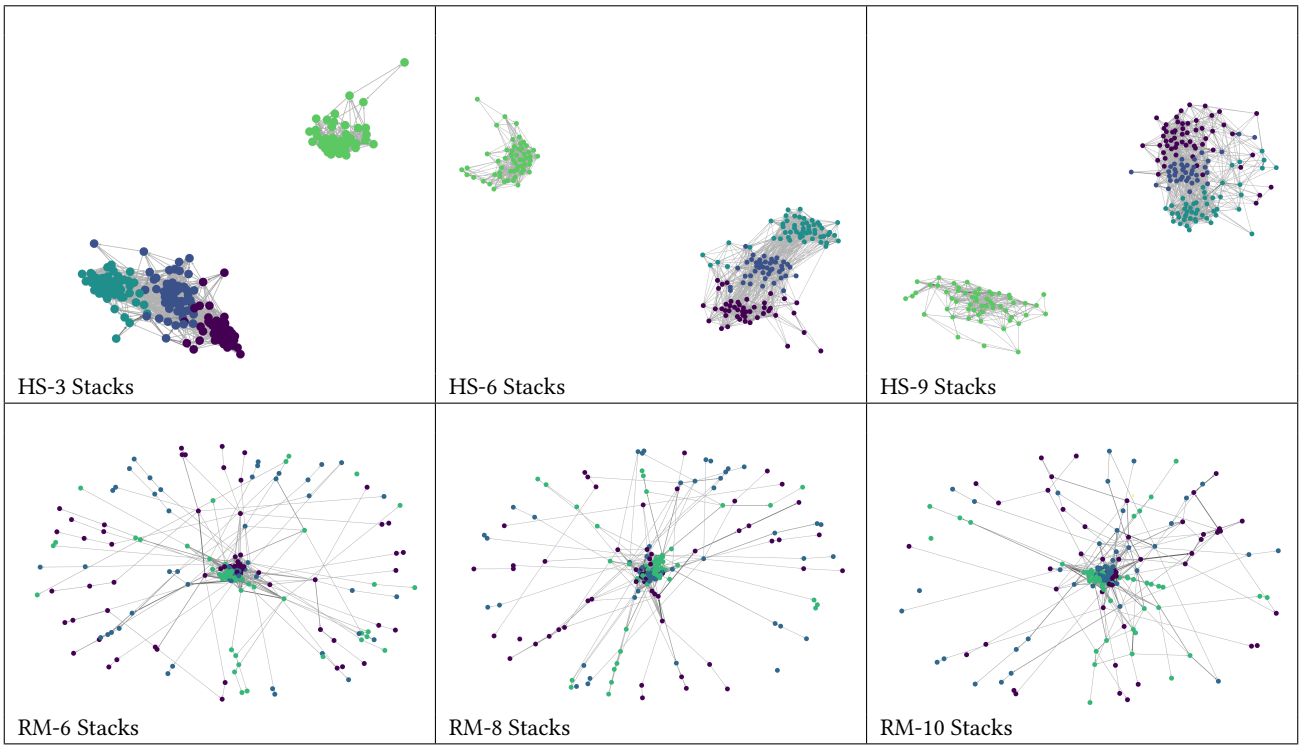

Figure 4: System State Transitions - Spring layout graphs of the states (nodes) and transition probabilities (edges). Thicker, darker edges imply more usual state transitions. Node colors correspond to the most frequent dynamic parameter setting with its corresponding cluster.

[11] Stuart A Kauffman and Edward D Weinberger. 1989. The NK model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of theoretical biology* 141, 2 (1989), 211–245.

[12] Moo Ho Lee, Chonghun Han, and Kun Soo Chang. 1999. Dynamic optimization of a continuous polymer reactor using a modified differential evolution algorithm. *Industrial & Engineering Chemistry Research* 38, 12 (1999), 4825–4831.

[13] Hui Lu, Jinhua Shi, Zongming Fei, Qianlin Zhou, and Kefei Mao. 2017. Measures in the time and frequency domains for fitness landscape analysis of dynamic optimization problems. *Applied Soft Computing* 51 (2017), 192–208.

[14] Katherine M Malan and Andries P Engelbrecht. 2013. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences* 241 (2013), 148–163.

[15] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke. 2012. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation* 6 (2012), 1–24.

[16] Gabriela Ochoa and Katherine Malan. 2019. Recent advances in fitness landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1077–1094.

[17] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. 2014. Local optima networks: A new model of combinatorial fitness landscapes. In *Recent advances in the theory and application of fitness landscapes*. Springer, 233–262.

[18] Djamila Ouelhadj and Sanja Petrovic. 2009. A survey of dynamic scheduling in manufacturing systems. *Journal of scheduling* 12, 4 (2009), 417.

[19] Ben Peterson, Iiro Harjunkoski, Samid Hoda, and John N Hooker. 2014. Scheduling multiple factory cranes on a common track. *Computers & Operations Research* 48 (2014), 102–112.

[20] Victor Pillac, Michel Gendreau, Christelle Guéret, and Andrés L Medaglia. 2013. A review of dynamic vehicle routing problems. *European Journal of Operational Research* 225, 1 (2013), 1–11.

[21] Erik Pitzer and Michael Affenzeller. 2012. A comprehensive survey on fitness landscape analysis. *Recent advances in intelligent engineering systems* (2012), 161–191.

[22] Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. 2016. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research* 54, 1 (2016), 215–231.

[23] Philipp Rohlfshagen and Xin Yao. 2009. The dynamic knapsack problem revisited: A new benchmark problem for dynamic combinatorial optimisation. In *Workshops on Applications of Evolutionary Computation*. Springer, 745–754.

[24] Philipp Rohlfshagen and Xin Yao. 2013. Dynamic combinatorial optimization problems: A fitness landscape analysis. In *Metaheuristics for Dynamic Optimization*. Springer, 79–97.

[25] Renato Tinós and Shengxiang Yang. 2014. Analysis of fitness landscape modifications in evolutionary dynamic optimization. *Information Sciences* 282 (2014), 214–236.

[26] Laurens Van Der Maaten. 2014. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research* 15, 1 (2014), 3221–3245.

[27] Mang Wang, Bin Li, Guofu Zhang, and Xin Yao. 2017. Population evolvability: Dynamic fitness landscape analysis for population-based metaheuristic algorithms. *IEEE Transactions on Evolutionary Computation* 22, 4 (2017), 550–563.

[28] Edward Weinberger. 1990. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological cybernetics* 63, 5 (1990), 325–336.

[29] Shengxiang Yang and Xin Yao. 2005. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft computing* 9, 11 (2005), 815–834.

[30] Elisabeth Zehendner, Dominique Feillet, and Patrick Jaillet. 2017. An algorithm with performance guarantee for the online container relocation problem. *European Journal of Operational Research* 259, 1 (2017), 48–62.

Measure	Type	Description
StorageSpaces	constant	Number of storage spaces in the warehouse
Stacks	constant	Number of stacks
Cranes	constant	Number of cranes
MaxStackHeight	constant	Maximum capacity of any stack in the warehouse
MeanPossibleCranes	constant	How many cranes can reach a stack on average
StoredBlocks	current	Number of occupied storage spaces
UsedStacks	current	Number of stacks with at least one stored block
MaxUsedHeight	current	Maximum height of current stacks
OutOfOrderDegree	current	Number of block pairs in the same stack stored in reversed handover order
HandoverMissingCount	current	Number of blocks in the rolling mill program but not in the system
OverdueBlocks	current	Number of stored blocks that are overdue
WeightedDistanceToHandover	current	Sum of all distances d_b from stored blocks $b \in B$ location to the handover stack, weighted by $\frac{1}{s_b}$, where s_b is the block's sequence number
AvailableForHandover	current	Number of blocks that can be put on the handover stack without intermediary moves (transshipments)
ReadyForHandover	current	Number of blocks that are ready for handover
ReadyAndAvailableForHandover	current	Number of blocks that are ready for handover and can be put on the handover stack without intermediary moves (transshipments)
MeanDueness	current	Mean time before due date
MaxDueness	current	Time to latest known due date
MinDueness	current	Time to earliest known due date
StdDueness	current	Standard deviation of times to due dates
StoredBlocks_Norm	current	StoredBlocks / StorageSpaces
UsedStacks_Norm	current	UsedStacks / Stacks
MaxUsedHeight_Norm	current	MaxUsedHeight / MaxStackHeight
MeanUsedHeight	current	StoredBlocks / Stacks
MeanUsedHeight_Norm	current	MeanUsedHeight / MaxStackHeight
OutOfOrderDegree_Norm	current	OutOfOrderDegree divided by maximum OutOfOrderDegree given current StoredBlocks
HandoverMissingCount_Norm	current	HandoverMissingCount / (HandoverMissingCount + StoredBlocks)
OverdueBlocks_Norm	current	OverdueBlocks divided by number of blocks for which due date is known
WeightedDistanceToHandover_Norm	current	WeightedDistanceToHandover divided by the warehouse's width
AvailableForHandover_Norm	current	AvailableForHandover / StoredBlocks
ReadyForHandover_Norm	current	ReadyForHandover / StoredBlocks
ReadyAndAvailableForHandover_Norm	current	ReadyAndAvailableForHandover / StoredBlocks
BlockedTime	objective	Time the arrival stack is blocked
DeliveredBlocks	objective	Number of blocks successfully handed over
CraneManipulations	objective	Number of executed crane moves
Overdueness	objective	Cumulative delay of all blocks
Messups	objective	Number of incorrectly delivered blocks
BlockedTimeRate	dynamic	Change of BlockedTime in the last n seconds, divided by n
DeliveredBlocksRate	dynamic	Change of DeliveredBlocks in the last n seconds, divided by n
CraneManipulationsRate	dynamic	Change of CraneManipulations in the last n seconds, divided by n
OverduenessRate	dynamic	Change of Overdueness in the last n seconds, divided by n
MessupsRate	dynamic	Change of Messups in the last n seconds, divided by n
Change Severity	dynamic FLA	see Section 3.3

Table 1: Features comprising the state descriptors.