AN EFFICIENT HEURISTIC FOR MULTI-OBJECTIVE TRAIN LOAD PLANNING: A PARAMETER SENSITIVITY ANALYSIS

Hilde Heggen[a], Kris Braekers[b], An Caris[c]

[a],[b],[c]UHasselt, Research group Logistics, Agoralaan, 3590 Diepenbeek, Belgium
[a]Maastricht University, Department of Quantitative Economics, P.O. Box 616, 6200 MD Maastricht, the Netherlands
[b]Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussel, Belgium

[a]hilde.heggen@uhasselt.be, [b]kris.braekers@uhasselt.be , [c]an.caris@uhasselt.be

ABSTRACT
Intermodal transportation is evolving towards a synchromodal system with real-time switching, in which most assignment decisions are postponed until the final planning phase. In such environment, it is important to assist train planners by providing decision support in the operational environment in which they operate. One important task for train planners concerns train load planning, which is concerned with the assignment of load units to the available slots on a train. In this paper, a sensitivity analysis of the parameters of a multi-objective heuristic algorithm for train load planning is presented, which accounts for different aspects of train capacity utilization. The tuned heuristic algorithm provides a number of load plans from which planners can choose the most relevant plan for the specific circumstances they are operating in at that moment.

Keywords: train load planning, intermodal transportation, multi-directional local search

1. INTRODUCTION
The aim of train load planning is to find an assignment of load units to the available locations or slots on a train. The objective can be based on either capacity utilization or handling operations at the intermodal terminal. At the same time, axle payloads, wagon weights and total train weights are restricted and a balance should exist between the payloads on the adjacent axles of each wagon. A major contribution to the development of the train load planning problem has been provided by Corry and Kozan (2008), who developed a first realistic model. Furthermore, Bruns and Knust (2012) are the first to adopt continuous weight restrictions in a train load planning problem.
Current literature often presents linear programming problems and solves them using commercial software. Heuristic solution methods are usually based on a local search, sometimes combined with simulated annealing. Commonly used neighbourhoods are a load unit swap and a configuration change for two wagons. These heuristic methods are mainly applied to problems in which the load units to be loaded are fixed and known, and only the final assignment to specific locations on the train must be determined. The actual load unit's location on the train is uncertain because not all information about handling operations at the terminal is known in advance and a slot may still be occupied at the moment of arrival of a load unit assigned to that slot. In these cases, it becomes a problem with a rolling horizon in which the current load plan serves as initial solution and new events trigger a local search.
Recently, the optimization of train load planning has been integrated with optimization of other operational decisions in an intermodal seaport terminal. The topic is first introduced by Ambrosino et al. in 2011, who combine it with the optimization of crane and storage planning. A number of linear programs have been solved for this type of problem. Moreover, a combination of a primal heuristic with a RANS matheuristic (Anghinolfi and Paolucci 2014) and a GRASP (Anghinolfi et al. 2014) have been proposed.
In this paper, we focus on various aspects of train capacity utilization. During train load planning, the planning department is responsible for managing all transport orders, assigning them to the right transport route, and in a second phase the assignment of these orders to the available locations on a train. Especially with the rising importance of synchromodality, which is associated with a dynamic process and real-time switching, and more load units available than the number of slots on a given train, planners are facing a complex decision process. They receive a lot of information and should decide on the most appropriate load plan using this information.
Although planners in real life should take many objectives into account, to the best of our knowledge, only one paper (Ambrosino et al. 2016) applies a multi-objective approach, comparing three exact approaches to solve the train load planning problem in seaport terminals, hereby focusing on operations in the crane and storage area. We propose a multi-directional local search heuristic focusing on a number of capacity-related objectives which train planners take into account during their planning process in Section 2. Moreover, the heuristic parameters are tuned (Section 3) and a sensitivity analysis is performed (Section 4). Finally, Section 5 presents the main conclusions.

2. HEURISTIC FRAMEWORK

We extend the case of Heggen et al. (2016), who consider a real-life case study of a network operator which owns and manages its own trains, to a multi-objective problem and solve it using a multi-directional local search (MDLS) heuristic. The aim is to find a configuration for each wagon, respecting axle payload, wagon weight and total train weight limits, while preserving a balance between the payloads on adjacent wagon axles. Furthermore, trains stop at an intermediate terminal and some wagons are decoupled before arriving at the final destination terminal due to a more restrictive path weight in the last part of the itinerary. The proposed multi-objective heuristic to solve the problem (Section 2.1) and its components (Section 2.2-2.5) are described, as well as indicators for the heuristic quality assessment (Section 2.6).

2.1. Multi-Directional Local Search Heuristic

The multi-directional local search heuristic framework of Tricoire (2012) is used to solve the train load planning problem with three objectives to be maximized:

1. Length utilization;
2. Destination preference scores for assigning load units to wagons unloaded at a more preferred unload terminal;
3. Priority scores for more urgent load units.

The method relies on the knowledge that it is sufficient to search in the direction of each of the objectives individually to find new, non-dominated solutions (Braekers et al. 2016). Consequently, single-objective operators can be implemented in the framework. Other advantages include the flexibility and simplicity of the method. A global pool of non-dominated solutions *E* is maintained and updated during the search. In an MDLS-iteration, a solution is selected randomly from the solution pool *E*. Then, starting from this solution, a distinct local search for each objective is performed.

For the train load planning problem, an initial solution is generated in a first constructive phase. Next, in the MDLS-framework, the local search operators are defined by altering the configuration of two wagons. For each objective specifically, wagons are selected in a different way, and a distinct acceptance criterion is used. Each local search ends if a maximum number of consecutive iterations without improvement is reached.

2.2. Constructive Phase

An initial solution is constructed by assigning load units to slots on each wagon, one by one, going from the front to the back of the train, using an intelligent candidate list to select load units first based on the highest priority, then highest weight. First, only critical load units are considered. Only after these are feasibly assigned, the remaining load units are considered. For the wagon under consideration, configurations which are more preferred with respect to the available length used are selected first. The available slots of the selected configuration are filled with load units matching the slot dimensions as long as the bogie, wagon and train weight limits are respected. If either not all slots in a configuration can be filled with the remaining available load units or the bogie balance limits are not respected, a next configuration is selected. Otherwise, all slots in the configuration are filled and the assignment procedure continues with the next wagon. This constructive phase results in a single initial solution, which is added to solution pool *E*. No randomness is involved at this point to avoid ending up with critical load units not being assigned.

2.3. Local Search Operators

Next, $n_{it(MDLS)}$ iterations of the MDLS heuristic are performed. One MDLS-iteration consists of three local searches on a single randomly selected solution $s \in E$. Each local search LS_k guides the search primarily towards improving one objective *k*. The neighbourhood is defined by simultaneously altering the configuration for two wagons, i.e. assigning a new configuration to these wagons. The way in which wagons are selected differs depending on the main objective focus of the local search, as discussed in Section 2.5. All load units which were assigned to the two selected wagons are added to the pool of available, currently unassigned load units. Next, configurations for both wagons are selected randomly with a higher probability to be selected if a configuration uses more wagon length. The probability of selecting a configuration is determined by the contribution of the length used in that specific configuration compared to the total length of all possible configurations for one wagon type. In this way, the probability of rejecting a solution because it does not satisfy the acceptance criterion, is reduced. Finally, the selected configurations are fixed for both wagons only if all critical load units can be assigned, dimensions of the selected load units match the slot dimensions and all constraints related to train, wagon and bogie weight limits as well as the bogie balancing are satisfied.

2.4. Evaluation of the Solutions

While other MDLS-approaches use pure single-objective local search procedures, our operators are guided by a normalized, weighted-sum objective function which takes into account all three objectives. This function assigns a weight w_k to the primary objective *k*, while the remaining objectives each receive a weight of $w_r = (1-w_k)/2$ (with $w_r << w_k$) in order to avoid a large negative change in these remaining objectives. Further, a temporary set of non-dominated solutions *T* is updated with new solutions within one local search. If a solution is non-dominated by the solutions in the temporary set, it is added to this set, while dominated solutions are removed. Working with this temporary solution set avoids updating solution set *E* too often when new solutions are found within one local search, especially because one local search primarily focuses on one objective only. Finally, the local search ends with updating the global archive of non-dominated solutions *E* with the temporary set of solutions *T* obtained in LS_k.

2.5. Local Search for each Objective

For the local search in the direction of improving the length utilization, two wagons are selected as follows: the first wagon is chosen randomly from all wagons for which not all loading length is utilized; the second wagon is selected purely random from all available wagons. Finally, the local search procedure is terminated if a maximum number of sequential iterations without improvement, or a solution with the maximum possible length utilization is reached. The search in the direction of improving destination preferences and priority scores is guided by a local search which consists of a similar configuration change operator. However, both neighbourhoods are defined by assigning a new configuration to two randomly selected wagons.

2.6. Quality Indicators

In order to assess the performance of the MDLS, two complementary quality indicators are introduced. First, the hypervolume indicator I_H measures the hypervolume covered by a set of solutions relative to a reference point. Second, the (multiplicative) epsilon indicator I_ε determines the factor by which each point in an approximation set (obtained by the heuristic algorithm) should be multiplied such that a reference set, which ideally consists of the exact Pareto-front, is weakly dominated by the approximation set (Knowles et al. 2006). The closer both indicators are to one, the better the quality of the approximation set. In order to compare the approximation sets obtained by two variants of a heuristic design, the two quality indicators can be used together, as each indicator measures slightly different information. Furthermore, if the indicators show opposite preference, the sets can be considered incomparable.

Both indicators are visualized in Figure 1 for a bi-objective maximization problem. The left-hand side shows the hypervolumes covered by a reference set and an approximation set. On the right-hand side, the crosses indicate the weakly dominated set obtained by the epsilon indicator.

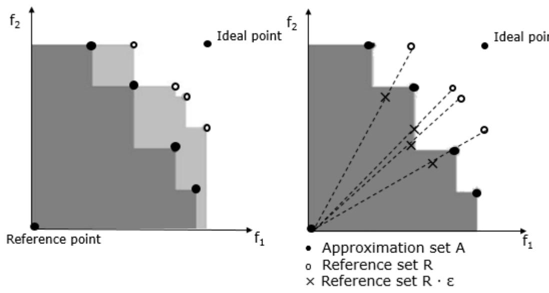

Figure 1: Quality Indicators (Adapted from Parragh et al. (2009))

In this paper, the hypervolume indicator I_H is stated as the hypervolume of the reference set which is covered by the approximation set generated by the MDLS (I_{Ha}/I_{Hr}). Moreover, both indicators are calculated after normalization of the objective values to ensure that each objective contributes more or less equally. The minimum reference point is determined based on the characteristics of the critical load units: the urgency score and length of the critical load units of an instance minus one and a minimum destination preference score of zero.

3. PARAMETER TUNING

irace (López-Ibáñez et al. 2016), a promising iterated racing procedure for tuning algorithm parameters, is used in order to find a parameter setting which leads to an excellent heuristic performance, obtaining solutions of good quality, such that the heuristic can be used as a decision support tool by practitioners. First, the tuning instances (Section 3.1) and considered MDLS-parameters (Section 3.2) are presented. Next, the iterated racing procedure for the multi-objective train load planning problem is described and the parameters are tuned (Section 3.3).

3.1. Instance Classes

A heterogeneous set of instance classes is used as input. Sets of load units with weights (in tonnes) being either light with *TRIA(17,20,23)*, heavy with *TRIA(23,26,29)* or uniformly distributed with *UNIF(17,29)* are considered. Furthermore, the number of critical load units is varied and can be 35% or 20% of the total amount of available load units. These six instance classes are applied to a wagon set of 5, 10 and 20 wagons, resulting in 18 classes.

3.2. Parameters of the MDLS

Two important parameters for the MDLS are the number of times a new solution is selected from the pool of non-dominated solutions, and the number of consecutive iterations without improvement after which the local search phase in the direction of each objective ends. Clearly, a trade-off between the values of these parameters can be expected if a limited computation time is available. In this section, no limit on the computation time is considered, but the relationship between solution quality and computation time for different amounts of MDLS-iterations and LS-iterations are examined in Section 4.

Within the scope of a single local search LS_k, the focus when accepting new solutions is on the main objective k. The weight w_k attached to the main objective of a local search should be tuned carefully. Moreover, within the normalized, weighted objective function used for accepting new solutions, possible criteria for accepting temporary solutions to continue working with within each local search A_{LS_k} (i.e. accepting worse, equal or better solution values for main objective k) are evaluated. Independent of this, only non-dominated, accepted solutions are added to the solution pool.

Finally, it is tested if the local search in the direction of improving destination preferences performs better when changing the configuration of two wagons with different destinations. An overview of the heuristic parameters under consideration, as well as its considered range of values are presented in Table 1.

Table 1: Tuning Parameters

Parameter	Description	Range
$n_{it(MDLS)}$	# times a random solution is selected	(10, 1000)
$n_{it(LS\_Length)}$	# consecutive non-improving iterations for LS_{Length}	(100, 2000)
$n_{it(LS\_Urgency)}$	# consecutive non-improving iterations for $LS_{urgency}$	(100, 2000)
$n_{it(LS\_DP)}$	# consecutive non-improving iterations for LS_{DP}	(100, 2000)
w_k	Weight attached to the main objective k in LS_k	(0.8, 1)
A_{LS_Length}	Accepting solutions with main objective ≥ CurrentBest + A_{LS_Length}	(-45,-20,0,20)
$A_{LS\_Urgency}$	Accepting solutions with main objective ≥ CurrentBest+ $A_{LS\_Urgency}$	(-2, 2)
A_{LS_DP}	Accepting solutions with main objective ≥ CurrentBest + A_{LS_DP}	(-2, 2)
$DP_{wgndestin}$	Select two wagons with different destinations in LS_{DP}	(0, 1)

The acceptance criterion in the local search for the length utilization considers only the specified ordinal values, while for the other parameters values between the specified minimum and maximum bounds can be selected by the racing procedure. The weight attached to the principal objective is a real number with a precision of two decimals, all other parameter values are integers. The ranges of all parameters are defined after preliminary testing and based on knowledge about the problem characteristics.

3.3. Iterated Racing Procedure

The iterated racing procedure (Figure 2) is able to automatically configure algorithms, providing a set of parameter values which performs well for a particular problem (López-Ibáñez et al. 2016).

```
Require: $I = \{I_1, I_2, \ldots\} \sim \mathscr{I}$,
    parameter space: $X$,
    cost measure: $C(\theta, i) \in \mathbb{R}$,
    tuning budget: $B$
 1: $\Theta_1 \sim SampleUniform(X)$
 2: $\Theta^{elite} := Race(\Theta_1, B_1)$
 3: $j := 1$
 4: while $B_{used} \leq B$ do
 5:     $j := j + 1$
 6:     $\Theta^{new} \sim Sample(X, \Theta^{elite})$
 7:     $\Theta_j := \Theta^{new} \cup \Theta^{elite}$
 8:     $\Theta^{elite} := Race(\Theta_j, B_j)$
 9: end while
10: Output: $\Theta^{elite}$
```

Figure 2: Iterated Racing Procedure (López-Ibáñez et al. 2016)

The racing procedure starts with T^{first} instances on which a number of uniformly sampled candidate parameter configurations are tested. After these T^{first} tested instances, the candidate configurations which perform worse than at least one other configuration – calculated by a statistical Friedman test – are discarded (line 1-2). The best configurations (i.e. the configurations with the best objective values) are selected as an elite set, and new configurations are added for the following race based on well-performing parent elite configurations found so far (line 5-8). In the next iterations or races, each time T^{each} instances are evaluated before discarding any configuration. Furthermore, the standard deviation is reduced for each parameter as the number of iterations increases in order to search closer around better performing values. The procedure is terminated if a predefined computational budget B is reached. This budget corresponds to a maximum number of experiments, where one experiment consists of one parameter configuration tested on a single instance.

As multiple objectives must be considered, the cost function is represented by the quality indicator value, which should be maximized. López-Ibáñez et al. (2016) tested *irace* for their problem with multiple objectives using the hypervolume and the epsilon indicator and could not find significant differences. Therefore, only the hypervolume indicator is used as measure of the solution quality at this stage. The calculation of this quality indicator requires a reference set, which can be the exact Pareto-front. If not all Pareto-optimal solutions are known, the reference set consists of all non-dominated solutions found by a number of MDLS-runs, combined with the non-dominated solutions found so far in the exact procedure. Therefore, all candidate parameter configurations in a single iteration are tested on one instance before evaluating the cost function, i.e. calculating the quality indicator. Normalization bounds and the reference point can be calculated in advance, independent of the approximation sets found by the heuristic.

Table 2 shows the adapted *irace* parameters used. All other parameters are at their default values.

Table 2: *irace* Parameters

***Irace* parameter**	**Value**
Tuning budget B	5000
Cost measure C	Hypervolume
T^{first}	36
T^{each}	18
Random samples	Off

The total set of tuning instances consists of two blocks of 18 instances, with a representative set of characteristics. The total amount of 36 instances, containing two instances from each instance class, is first tested before discarding any candidate configuration. In this way, two instances of every class are evaluated before a first elimination occurs, to cope with a possible outlier instance. Next, after every block of 18 instances, the configurations under consideration are again evaluated. Sampling of instances does not occur randomly, but in the order of the instance classes within one block in order to avoid elite configurations being biased towards only a subset of the instance classes.

The best configurations presented by *irace* are summarised in Table 3. These configurations are ordered according to their mean performance, but do not show a statistically significant difference with respect to the solution quality. The average hypervolume indicator value of the best configuration across all considered instances amounts to 0.9996.

Table 3: Best Configurations

Parameter	C_1	C_2	C_3	C_4
$n_{it(MDLS)}$	987	949	759	775
$n_{it(LS\_Length)}$	801	377	434	271
$n_{it(LS\_Urgency)}$	1826	1498	1745	1384
$n_{it(LS\_DP)}$	1716	1541	1388	1631
w_k	0.91	0.92	0.90	0.90
A_{LS_Length}	0	0	0	0
$A_{LS\_Urgency}$	0	0	0	0
A_{LS_DP}	0	0	0	0
$DP_{wgndestin}$	0	0	0	0

Differences exists with respect to the number of iterations $n_{it(MDLS)}$ and $n_{it(LS\_k)}$, while the acceptance thresholds are identical for all best configurations C_1 to C_4. The obtained parameter value for A_{LS_DP} indicates that solutions are accepted only if they are better than the current best solution within a single local search. The acceptance threshold within the other two local searches is defined differently and therefore, the obtained parameter values indicate that solutions are accepted if they are at least as good as the current best solution. One important limitation of *irace* is that the automatic algorithm configuration does not take into account computation times when selecting parameter configurations. Therefore, these results should be further tested and a sensitivity analysis will be performed on the parameters to analyse differences in computation times and solution quality for different parameter settings.

4. SENSITIVITY ANALYSIS

The best parameter configuration C_1 resulting from the *irace* tuning procedure is analysed to examine the influence of changes in these parameter values on solution quality and computation times. For this purpose, two new test instances per class are used, which are not considered in the tuning phase. Varying parameter values are tested on these instances with respect to differences in solution quality, expressed as a proportional deviation from the hypervolume of the reference set (HV_R), as well as differences in computation time. First, interactions between $n_{it(MDLS)}$ and $n_{it(LS\_k)}$ are considered (Section 4.1). Next, it is examined whether a temporary solution pool T, maintained within a single local search, influences computation times (Section 4.2). Finally, variations in parameter values w_k, A_{LS_k} and the possibility of selecting two wagons with different destinations ($DP_{wgndestin}$) in LS_{DP} are tested (Section 4.3). The main findings are summarised in Section 4.4.

4.1. Interaction between $n_{it(MDLS)}$ and $n_{it(LS\_k)}$

It can be expected that a higher number of iterations, both $n_{it(MDLS)}$ and $n_{it(LS\_k)}$, corresponds to a higher solution quality, but at the cost of larger computation times. Therefore, a point may be determined as from which additional gains in solution quality become small relative to the increase in computation time. Values up to 1000 MDLS-iterations $n_{it(MDLS)}$ are considered with steps of 200 iterations. As the largest gains may be obtained during the first MDLS-iterations, 10, 50 and 100 iterations are added. To examine its interaction with the number of consecutive iterations without improvement $n_{it(LS\_k)}$ after which each local search is ended, for each local search LS_k separately multiples of 250 consecutive non-improving iterations are considered with a maximum of 2000 iterations.

Figure 3 shows the average proportional deviation from the hypervolume of the reference set when either varying the number of LS-iterations for destination preferences (*DP*), length utilization (*Length*) or urgency scores (*Urgency*). The remaining parameters are set at the values of the best-performing configuration. Variations in the number of non-improving LS-iterations after which the local search with respect to the length utilization ends do not significantly influence solution quality. This corresponds to the relatively small parameter values for $n_{it(LS\_Length)}$ in the best *irace* configurations. For the destination preference scores and urgency scores, larger differences can be observed for low numbers of MDLS-iterations. Clearly, major improvements with respect to the solution quality are reached during the first MDLS-iterations. These results are consistent with the best configurations found by *irace*, as $n_{it(LS\_DP)}$ and $n_{it(LS\_Urgency)}$ are always larger than 1250.

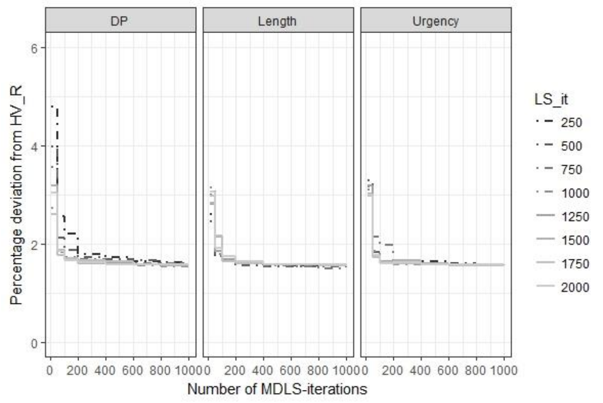

Figure 3: Solution Quality Based on $n_{it(MDLS)}$ and $n_{it(LS\_k)}$

Additionally, Figure 4 displays the average solution quality depending on the number of MDLS-iterations over all experiments, grouped per instance size. These results show that a clear difference exists with respect to the average solution quality: the heuristic performance is highest for instances with 10 and 20 wagons, while the mean performance is lower for 5 wagons. However, the mean deviation is still lower than 5%.

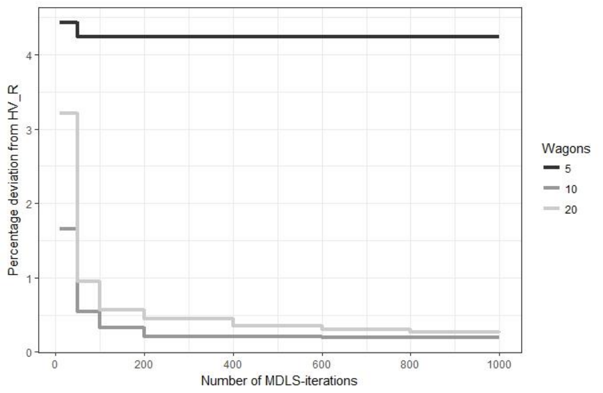

Figure 4: Average Solution Quality per Instance Size

Generally, for low numbers of MDLS-iterations, differences in solution quality are larger between a high and low number of non-improving local search iterations. As from 400 MDLS-iterations, the improvement in solution quality becomes relatively small, independent of the number of non-improving local search iterations.

Figure 5 shows average computation times in seconds for all instances of each possible combination of MDLS- and LS-iterations. The number of MDLS-iterations mainly influences computation time, as it involves additional iterations for all three local searches at the same time. The difference in the slope of the destination preference graph compared to the length and urgency graphs indicates that each LS_{Length} and $LS_{urgency}$ reached the number of consecutive iterations without improvement earlier, ending the local search. For LS_{DP}, this implies that improvements are found at a later stage of the respective local search, initializing the search again with a new best solution without ending the local search.

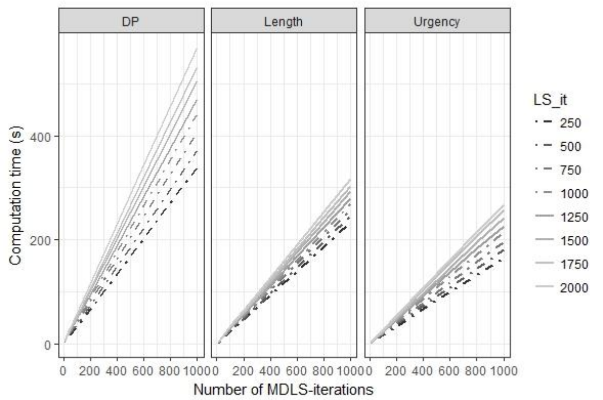

Figure 5: Average Computation Times Based on $n_{it(MDLS)}$ and $n_{it(LS\_k)}$

Computation time primarily rises with the number of MDLS-iterations, while solution quality remains relatively stable beyond 400 MDLS-iterations, regardless of the number of LS-iterations. For the best configurations C_1-C_4 the number of iterations for LS_{Length} varies between 250 and 1000, and for $LS_{Urgency}$ and LS_{DP} between 1250 and 2000. For these intervals of LS-iterations, the solution quality is high and stable as from 200 MDLS-iterations with a short computation time compared to 400 MDLS-iterations. Based on these observations, for the remainder of this paper we use $n_{it(MDLS)}$ = 200, while $n_{it(LS\_k)}$ remains at the best values found by *irace*. This configuration leads to high-quality solutions in relatively short computation times.

4.2. Effect of a Temporary Solution Pool T

Next, the effect of a temporary non-dominated solution pool *T*, used within the scope of a single local search, on computation times is evaluated by means of a paired-samples *t*-test. The temporary solution pool does not influence the solution quality. However, it may impact total computation times. For each instance category, average computation times as well as the *p*-values are displayed in Table 4.

Table 4: Effect of *T* on Computation Time (s)

	Temporary pool		
Instance class	**No**	**Yes**	***p*-value**
(5, 35%, light)	53.53	53.34	0.159
(5, 35%, heavy)	322.67	320.59	*0.001
(5, 35%, unif)	67.92	67.66	0.081
(5, 20%, light)	72.66	72.62	0.911
(5, 20%, heavy)	218.72	218.84	0.070
(5, 20%, unif)	82.20	81.83	0.539
(10, 35%, light)	107.11	108.26	0.252
(10, 35%, heavy)	277.49	279.43	0.267
(10, 35%, unif)	174.63	169.89	0.239
(10, 20%, light)	35.05	35.21	0.726
(10, 20%, heavy)	19.83	20.14	0.055
(10, 20%, unif)	31.27	30.46	*0.035
(20, 35%, light)	129.78	130.38	0.784
(20, 35%, heavy)	136.85	135.47	0.714
(20, 35%, unif)	94.54	99.73	0.266
(20, 20%, light)	36.81	35.69	0.326
(20, 20%, heavy)	42.00	41.93	0.938
(20, 20%, unif)	31.66	31.76	0.828

Contrary to the expectations, we can conclude that, on the 5%-significance level, no statistically significant difference in computation time can be obtained by maintaining a temporary solution pool within a single local search, except for two instance classes (indicated with an asterisk). This may be explained by the fact that the number of non-dominated solutions in the global solution pool is relatively small. Moreover, during the first iterations, the temporary solution pool may provide advantages, as more new, non-dominated solutions are found and the obtained non-dominated solutions may be further away from the Pareto-front. However, as the number of iterations increases, it becomes harder to find new non-dominated solutions as these solutions are already close to the Pareto-front and the temporary solution pool remains relatively small. This implies that a smaller number of evaluations between the temporary pool *T* and the global pool *E* should be performed. Furthermore, computation times are relatively large for the smallest instances. This may partly be due to the fact that the heuristic performs additional iterations, even when the optimal Pareto-front already may have been reached. For larger instances, increased computation times are observed if a high number of critical load units is available.

Although no significant difference is observed for most instance classes, significantly lower computation times are observed for two instance classes if a temporary solution pool is included. Therefore, the temporary solution pool is maintained as a component of the MDLS-heuristic.

4.3. Sensitivity of w_k, A_{LS_k} and $DP_{wgndestin}$

For all other parameters under consideration, Table 5 (Appendix A) shows an overview of the average solution quality obtained for each instance size as well as the overall average solution quality for each parameter value. For most of the parameters, the sensitivity analysis shows results identical to the *irace* parameter configurations. However, for A_{LS_Length} the parameter value resulting in the overall average best result (indicated with an asterisk) does not correspond to the parameter values selected by *irace* (indicated in bold). This small deviation may be explained by the decision to work with 200 MDLS-iterations instead of a parameter value out of one of the best configurations as well as by the fact that different instances are used. In the remainder of this section, individual results for each parameter are discussed in detail.

With respect to the **weight** attached to the main objective of a single local search, w_k, values between 0.6 and 1 are tested with an interval of 0.05. The parameter value of 0.91 obtained in the best configuration C_1 is also added. The solution quality and computation times (in seconds) for each parameter value and each instance size (5, 10 and 20 wagons) are displayed in Figure 6 and Figure 7.

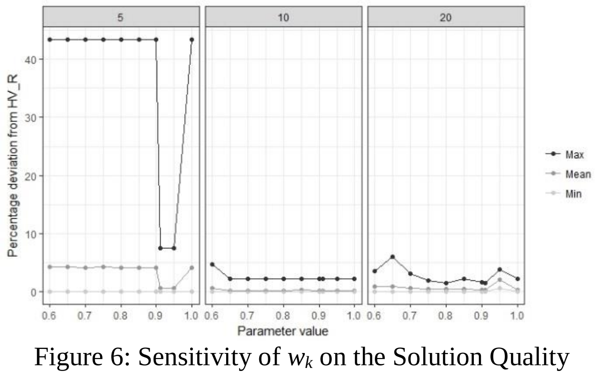

Figure 6: Sensitivity of w_k on the Solution Quality

Based on the results of the test instances used for these sensitivity analysis, for small instances with 5 wagons, the weight could be set to either 0.90 or 0.95. However, one outlier instance severely influences the average performance. For the larger and realistic instances with 10 and 20 wagons, differences in solution quality are smaller. Generally, a weight of 0.91 provides the highest average solution quality, while the lowest solution quality (with a high deviation from HV_R) is acceptable for all instances.

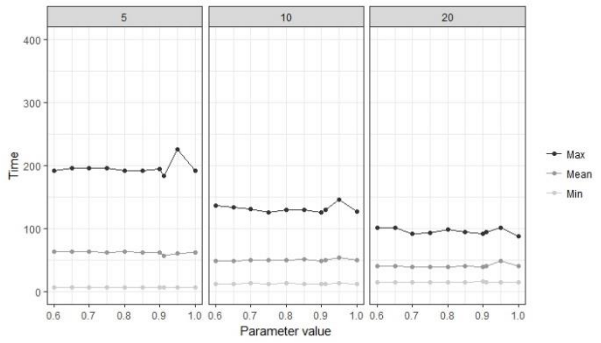

Figure 7: Computation Time (s) Based on w_k

Although computation times are consistent, average and maximum computation times show only small fluctuations, indicating that changes in this parameter do not substantially influence computation times.

Similarly, the influence of the criterion for **accepting solutions** in each local search LS_k for each objective k is evaluated. Figure 8 shows that the best parameter values for A_{LS_Length} are not consistent with the *irace* results for the considered test instances. This can be observed by the difference in solution quality between a small instance size of 5 wagons and larger instance sizes. As the performance for instances with 10 and 20 wagons is independent of the range of parameter values, accepting solutions with a length of 20 or 45 feet less than the current best solution provides a higher overall solution quality.

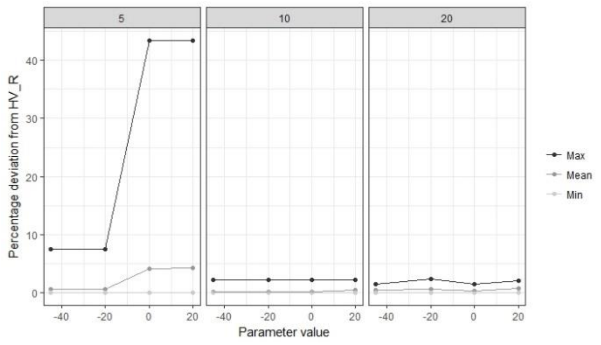

Figure 8: Sensitivity of A_{LS_Length} on the Solution Quality

Only small differences exist regarding average computation times, as demonstrated in Figure 9. The main difference exists for the smallest instance category, where maximum computation times rise, which may be due to the fact that the neighborhood is rather small when only accepting improving solutions. Considering computation times and solution quality simultaneously, allowing the acceptance of worse solutions might be favourable for these small instances.

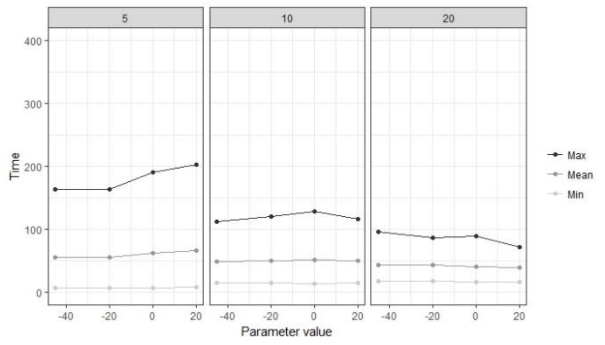

Figure 9: Computation Time (s) Based on A_{LS_Length}

With respect to $A_{LS\_Urgency}$, Figure 10 also shows a clear difference in solution quality for small instances in comparison with the instances of 10 and 20 wagons. However, at $A_{LS\_Urgency} = 0$, for the latter instance categories minimum and maximum solution quality are extremely close, and the overall average solution quality is the highest. This is compatible with the *irace* results.

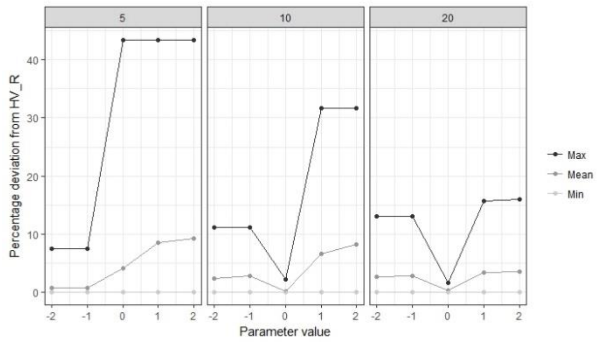

Figure 10: Sensitivity of $A_{LS\_Urgency}$ on the Solution Quality

Figure 11 displays computation times for each of the parameter values of $A_{LS\_Urgency}$. Although maximum computation times show a decreasing trend for instances with 10 and 20 wagons as the parameter value increases, average computation times show only a very weak decrease within each of the three instance sizes.

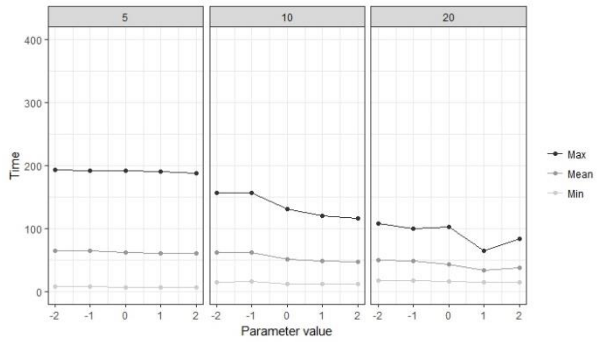

Figure 11: Computation time (s) Based on $A_{LS\_Urgency}$

Figure 12 shows for possible values of A_{LS_DP} a pattern similar to $A_{LS\_Urgency}$ with respect to the solution quality. Although instances with 5 wagons perform worse if $A_{LS\_Urgency} = 0$, the overall performance is highest.

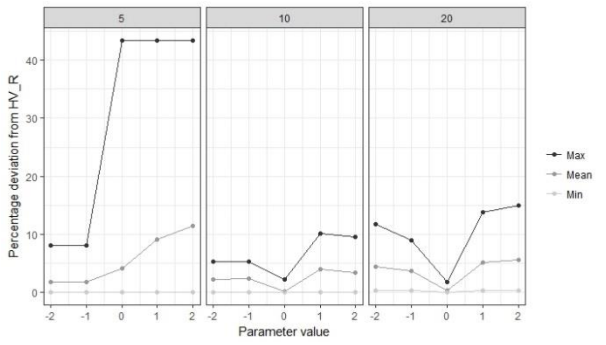

Figure 12: Sensitivity of A_{LS_DP} on the Solution Quality

As shown by Figure 13, only instances with 10 and 20 wagons show clear differences in computation time for different parameter values, especially for the maximum computation times.

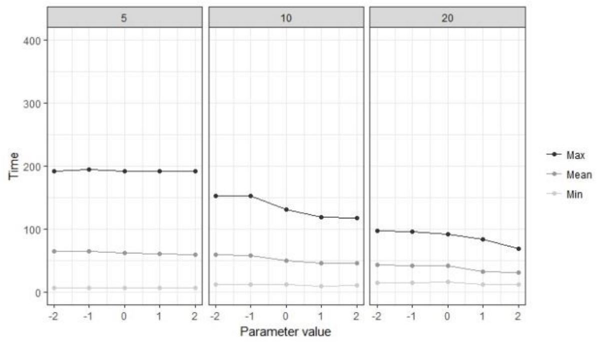

Figure 13: Computation Time (s) Based on A_{LS_DP}

Finally, a parameter $DP_{wgndestin}$ is added and evaluated in order to test whether in the local search focusing on destination preference scores, selecting two wagons with different **destinations** ($DP_{wgndestin} = 1$) leads to a higher solution quality in comparison with two random wagons ($DP_{wgndestin} = 0$). As shown in Figure 14, no substantial difference exists with respect to the solution quality. Moreover, the boxplots in Figure 15 indicate that maximum computation times mostly increase if in the local search is based on swapping two wagons with different destinations. Therefore, selecting two wagons with different destinations does not add value to the heuristic.

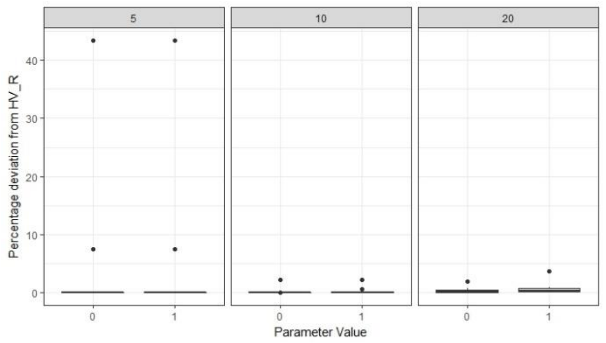

Figure 14: Sensitivity of $DP_{wgndestin}$ on the Solution Quality

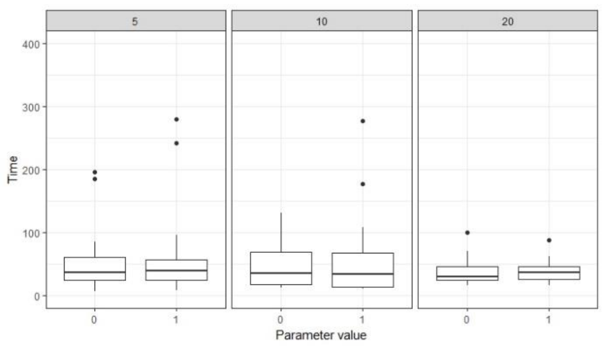

Figure 15: Computation Time (s) Based on $DP_{wgndestin}$

4.4. Practical Implications

Generally, the parameter configuration C_1 found by *irace* provides high-quality results, even with a smaller number of MDLS-iterations. However, one possible outlier instance in the instance category of 5 wagons might have influenced the results of the sensitivity analysis. Moreover, for that specific category a different configuration may be more suitable (e.g. in LS_{Length}, allowing to continue with solutions which are worse than the current best solution found so far), but the tuned parameter configuration presented by *irace* generally provides a reliable performance.

For practical applications it is important that good solutions are obtained in short computation times. Therefore, based on the discussed results, train load plans resulting from the MDLS using no more than 200 MDLS-iterations in order to reduce computation times will be valuable for practitioners. If a smaller number of MDLS-iterations would be considered, the influence of a temporary solution pool on computation times could be tested again.

5. CONCLUSIONS

In this paper, we were able to find a configuration which generally performs efficiently for a heterogeneous set of instance classes. With this parameter configuration, the multi-directional local search heuristic is able to find solutions of high quality within a reasonable amount of computation time. One limitation of this research is that only the hypervolume is used as a performance indicator. The obtained results may be validated with the results of the epsilon indicator. Further, the considered instances are heterogeneous with respect to their characteristics and maybe different parameter configurations would be selected if each category would be considered separately. Further research may focus on finding specific configurations for each category of instances, depending on the intended use of the heuristic. Moreover, while the interaction between the number of MDLS-iterations and the number of LS-iterations is investigated, no interaction effects are studied with respect to the parameters for $w_{k,}$, A_{LS_k}, $DP_{wgndestin}$.

The planning processes in intermodal transport are subject to many dynamics which influence the assignment decision. In this dynamic environment, the presented heuristic with the defined parameter setting can be used to provide decision support for planners in real-life planning contexts, while the final decision on the most appropriate load plan remains with the human planner.

ACKNOWLEDGMENTS

This work is partially funded by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (research project COMEX, Combinatorial Optimization: Metaheuristics & Exact Methods).

The computational resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government department EWI.

APPENDIX A

Table 5: Average Percentage Deviation from HV_R

Parameter	Value	Wagons			Overall
		5	**10**	**20**	
w_k	0.60	4.3306	0.6157	1.0049	1.9837
	0.65	4.3306	0.2217	0.9655	1.8393
	0.70	4.2386	0.2011	0.7245	1.7214
	0.75	4.2476	0.2095	0.5645	1.6739
	0.80	4.2386	0.2615	0.4667	1.6556
	0.85	4.2386	0.3367	0.4415	1.6723
	0.90	4.2386	0.1963	0.3564	1.5971
	0.91	**0.6272**	**0.2155**	**0.3957**	***0.4128**
	0.95	0.6272	0.2771	2.0622	0.9888
	1	4.2386	0.2001	0.3852	1.6080
A_{LS_Length}	-45	0.6272	0.2374	0.5776	*0.4807
	-20	0.6272	0.2269	0.6710	0.5084
	0	**4.2386**	**0.2073**	**0.4302**	**1.6254**
	20	4.3306	0.4452	0.8677	1.8812
A_{LS_Urg}	-2	0.7481	2.3843	2.7250	1.9525
	-1	0.7481	2.8099	2.8284	2.1288
	0	**4.2386**	**0.2215**	**0.3379**	***1.5993**
	1	8.5737	6.5884	3.4988	6.2203
	2	9.3047	8.3223	3.5622	7.0631
A_{LS_DP}	-2	1.8935	2.2498	4.4634	2.8689
	-1	1.8935	2.3854	3.7206	2.6665
	0	**4.2386**	**0.2105**	**0.3344**	***1.5945**
	1	9.1638	3.9724	5.1280	6.0881
	2	11.4811	3.3848	5.6054	6.8238
$DP_{wgndestin}$	**0**	**4.2386**	**0.1991**	**0.4045**	***1.6141**
	1	4.2386	0.3100	0.6993	1.7493

REFERENCES

Ambrosino D., Bramardi A., Pucciano M., Sacone S., Siri S., 2011. Modeling and solving the train load planning problem in seaport container terminals. Transportation Research Part E: Logistics and Transportation Review, 79, 65-82.

Ambrosino D., Bernocchi L., Siri, S., 2016. Multi-objective optimization for the train load planning problem with two cranes in seaport terminals. IFAC-PapersOnLine, 3 (49), 383-388.

Anghinolfi D., Caballini C., Sacone S., 2014. Optimizing train loading operations in innovative and automated container terminals. Proceedings of the 19th international federation of automatic control World Congress, pp. 9581-9586, August 24-29, Cape Town (South Africa).

Anghinolfi D., Paolucci M., 2014. A general purpose lagrangian heuristic applied to the train loading problem. Procedia - Social and Behavioral Sciences, 108, 37-46.

Braekers K., Hartl R.F., Parragh S.N., Tricoire F., 2016. A bi-objective home care scheduling problem: Analyzing the trande-off between costs and client inconvenience. European Journal of Operational Research, 248 (2), 428-443.

Bruns F., Knust S., 2012. Optimized load planning of trains in intermodal transportation. OR Spectrum, 34, 511-533.

Corry P., Kozan, E., 2008. Optimised loading patterns for intermodal trains. OR spectrum, 30 (4), 721-750.

Heggen H., Braekers K., Caris A., 2016. Optimizing Train Load Planning: Review and Decision Support for Train Planners. Proceedings of the 7th International Conference on Computational Logistics, pp. 193-208, September 7-9, Lisbon (Portugal).

Knowles J., Thiele L., Zitzler E., 2006. A tutorial on the performance assessment of stochastic multiobjective optimizers. TIK-Report No 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, revised version.

López-Ibáñez M., Dubois-Lacoste J., Pérez Cáceres L., Birattari M., Stützle T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives, 3, 43–58.

Parragh S.N., Doerner K.F., Hartl R.F., 2009. A Heuristic Two-Phase Solution Approach for the Multi-Objective Dial-A-Ride Problem. Networks, 54 (40), 227-242.

Tricoire F., 2012. Multi-directional local search. Computers & Operations Research, 39 (12), 3089-3101.

AUTHORS BIOGRAPHY

Hilde Heggen graduated in 2014 as a Master of Science in Business Engineering with a major in Operations Management and Logistics at Hasselt University, Belgium. In April 2015, she started her PhD focusing on train planning models in intermodal transportation.

Kris Braekers is Assistant Professor of Operations Management and Logistics at Hasselt University in Belgium since 2015. He obtained his PhD in Applied Economics at Hasselt University in 2012. His main research interest is studying combinatorial optimization problems in the context of transport and (healthcare) logistics (e.g., vehicle routing problems) and applying meta-heuristic techniques to solve these problems. Other research topics include warehouse management, empty container management and modelling intermodal freight transport networks.

An Caris was appointed in 2012 as Assistant Professor at the Faculty of Business Economics (BEW) of Hasselt University. She obtained her PhD in Applied Economics in 2010 and was first a postdoctoral research fellow of the Research Foundation - Flanders (FWO). Her research interest goes to the application of Operations Research (OR) techniques in the field of operations management and logistics. In her PhD thesis she focused on the competitiveness of intermodal transport making use of inland navigation. Currently she studies planning problems in warehouse operations, intermodal rail transport, collaborative logistics and healthcare logistics.