JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Fitness landscape analysis and design of metaheuristics for car sequencing

Uli Golle

Working Paper 03/2011
May 2011

Working Papers in Information Systems and Business Administration

Johannes Gutenberg-University Mainz
Department of Information Systems and Business Administration
D-55128 Mainz/Germany
Phone +49 6131 39 22734, Fax +49 6131 39 22185
E-Mail: sekretariat[at]wi.bwl.uni-mainz.de
Internet: http://wi.bwl.uni-mainz.de

Fitness landscape analysis and design of metaheuristics for car sequencing

Uli Golle

***Abstract*—We study the car sequencing (CS) problem, an NP-hard combinatorial optimization problem which aims at finding a production sequence of different models launched down a mixed-model assembly line. The models are differentiated by a number of selected options. For each option, a so-called sequencing rule is applied which restricts the option occurrences in the sequence in order to minimize the work overload of the respective line operators. In this paper, we perform a fitness landscape analysis of several instances of the CS problem, where we examine the autocorrelation as well as fitness-distance correlation (FDC) induced by four neighborhood operators and three distance metrics. These results are subsequently employed for the design of two metaheuristics for CS, a variable neighborhood search (VNS) and a memetic algorithm (MA) evaluated with three different crossover operators. The VNS shows superior performance and even improves currently best known solutions of instances in the CSPLib. Although the results of the MA algorithms are inferior compared to VNS, applying crossover operators that respect the adjacency distance metric lead to a better solution quality than using other crossovers.**

I. Introduction

The car sequencing (CS) problem [1] is a combinatorial optimization problem seeking a production sequence of various models launched down a mixed-model assembly line. The models are derived from a common base product and differ in a number of selected options. Nevertheless, all models are jointly manufactured on the same mixed-model assembly line with a lot size of one. Since the processing times of the models can vary and the line is balanced to an average model, a sequence of consecutive work-intensive models may lead to work overload of the respective line operators. Work overload occurs, whenever an operator can not finish his assigned tasks within the available station limits. These work overload scenarios have to be compensated by other strategies, such as employing additional utility workers or stopping the line. To minimize the total amount of work overload, CS uses a so-called sequencing rule of type $H : N$ for each selected option, which restricts the occurrence of models having this option to at most H, in any subsequence of N consecutive models. The aim is to find a sequence, which meets the demand for each model and satisfies all sequencing rules (constraint satisfaction problem) or minimizes the number of sequencing rule violations (optimization problem). CS belongs to the class of NP-hard problems [2].

The concept of landscapes is an intuitive notion of the search space, the set of all solutions to a combinatorial optimization problem. A search algorithm navigates through the landscape in order to find the best solution, usually the highest peak or the lowest valley. Therefore, all solutions have to be connected by a certain distance measure and the quality of each solution is assessed using a so-called fitness function. Thus, landscapes are often referred to as fitness landscapes. The analysis of fitness landscapes can give valuable insights into the characteristics of the search space. These informations can be employed to design better search algorithms that incorporate more problem-specific knowledge, which is crucial for effective optimization algorithms.

In this paper, we study the resulting landscapes of four neighborhood operators and three distance metrics for a set of CS instances. We analyze local and global properties of the landscapes by performing an autocorrelation and fitness distance correlation analysis. The results of the landscape analysis are employed to design two metaheuristics for CS, a simple variable neighborhood search (VNS) as well as a memetic algorithm (MA). Thereby, we propose a new heuristic crossover operator for CS. The performance of the VNS and the MA with the new crossover operator as well as two existing ones is evaluated in experiments using widely applied CS problem instances from the CSPLib. The main findings are:

1) Three out of the four neighborhood operators lead to a smooth landscape for CS in terms of the normalized correlation length.
2) All neighborhood operators and the adjacency distance metric show a high fitness-distance correlation on all instances and, thus according to [3], result in an 'easy' problem for evolutionary algorithms.
3) In case of the adjacency distance metric, a 'big valley' structure for CS is identified.
4) Incorporating the findings of the landscape analysis into operators for the VNS and the MA leads to high quality metaheuristics that find the best-known sequences for all test instances and even improve the currently best solution of two instances.

The remainder of the paper is organized as follows. In Section II, we review relevant literature on CS and state its optimization model. Section III presents four neighborhood operators proposed for CS and studies characteristics of the fitness landscapes resulting from these operators. The insights gained from Section III are used in Section IV to design a VNS as well as a MA for CS. The performance of the metaheuristics is evaluated in experiments in Section V. Section VI concludes the paper.

II. The car sequencing problem

The CS problem stems from applications in the automobile industry and was first introduced by [1]. As its related approach of Mixed-Model Sequencing (MMS) [4], it aims at finding a sequence of different models to be produced at

Table I
NOTATIONS

T	number of production slots (index t)
M	number of models (index m)
O	number of options (index o)
d_m	demand for model m
a_{om}	binary demand coefficient: 1, if model m requires option o, 0 otherwise
$H_o : N_o$	sequencing rule: at most H_o out of N_o successively sequenced models require option o
x_{mt}	binary variable: 1, if model m is produced in slot t, 0 otherwise
y_{ot}	binary variable: 1, if sequencing rule defined for option o is violated in window starting in cycle t
BI	Big Integer

a mixed-model assembly line with minimum work overload. Work overload occurs, whenever a line operator can not finish his assigned assembly tasks due to a consecutive order of work-load intensive models in the sequence. In contrast to MMS, CS applies a surrogate objective for the minimization of work overload. Given a pool of different models, which can be distinguished by selected binary options (such as having an air conditioning or not in case of car models), CS restricts the occurrences of each option $o \in O$ in the sequence by using so-called sequencing rules $H_o : N_o$. A sequencing rule allows only H_o models having option o in any subsequence of N_o consecutive models. For instance, a sequencing rule of 3:5 for the option 'sunroof' requires that at most 3 out of 5 consecutive models contain a sunroof. If more models have a sunroof, a violation of the respective sequencing rule occurs. The aim of CS is to find a production sequence containing all models in the pool and inducing the minimum overall number of rule violations. It is assumed that the minimization of sequencing rule violations simultaneously leads to the underlying goal of minimizing the total amount of work overload. With notations from Table I, the CS problem is formulated as an integer linear program as follows [5]:

$$\text{CS: Minimize} \quad \sum_{o \in O} \sum_{t=1}^{T-N_o+1} y_{ot} \tag{1}$$

$$\sum_{t=1}^{T} x_{mt} = d_m \quad \forall m \in M \tag{2}$$

$$\sum_{m \in M} x_{mt} = 1 \quad \forall t = 1, \ldots, T \tag{3}$$

$$\sum_{t'=t}^{t+N_o-1} \sum_{m \in M} x_{mt'} \cdot a_{mo} \leq H_o + y_{ot} \cdot BI \quad \forall o \in O; t = 1, \ldots, T - N_o + 1 \tag{4}$$

$$x_{mt} \in \{0,1\} \quad \forall m \in M; t = 1, \ldots, T \tag{5}$$

$$y_{ot} \in \{0,1\} \quad \forall o \in O; t = 1, \ldots, T \tag{6}$$

y_{ot} indicates whether a rule violation of option o occurs in a subsequence starting at position t. The objective function (1) minimizes the overall number of rule violations. The resulting sequence has to meet the required demand d_m for each model $m \in M$ (2) and is allowed to contain exactly one model at each production slot t (3). Constraints (4) check for rule violations. We apply the sliding-window approach, where a violation of a subsequence is always assessed with one violation in the objective function [5]. Finally, (5) and (6) ensure variables x_{mt} and y_{ot} to be binary.

Since CS is NP-hard in the strong sense [2], various exact, heuristic and hybrid search procedures have been developed in order to solve the problem [6], [7]. Among the exact approaches are integer linear programming (ILP) formulations. While [5] present the aforementioned basic ILP, the ILP by [8] focuses on option assignments to the sequence instead of models and also includes constraints of the preceding paint shop. [9] propose a scattered branch & bound algorithm introducing new lower bounds and dominance rules for CS. Their bounds are further improved by [10], who solve a graph representation of CS with an exact iterative beam search algorithm which is, according to experimental results, currently the best-known exact algorithm for CS.

As for heuristics and metaheuristics, different approaches including construction heuristics, local searches, evolutionary algorithms and ant colony optimizations have been developed. In [11], different greedy heuristics for the optimization version of CS are studied and applied in local search as well as ant colony algorithms. In [12] the constraint satisfaction version of CS is addressed and different value ordering heuristics to construct an optimal sequence are proposed. Different local search procedures are introduced in [13] including greedy as well as threshold accepting algorithms that employ a set of various neighborhood operators. A large neighborhood search is also introduced in [14] using different move operators and search strategies. In [15] and [16] an industrial version of CS with instances having more than 1000 models and including paint shop constraints and the distinction between hard and soft constraints, is solved by local search applying variable neighborhoods. Among evolutionary approaches, a genetic algorithm (GA) is developed in [17] using a version of adaptive template type crossover and performing hill climbing as mutation operator. For the industrial version of CS, a genetic local search procedure is designed in [18]. The authors introduce a crossover operator that preserves common subsequences in both parents and apply a local search using a shift neighborhood after each recombination. In [19] a standard GA is combined with squeaky-wheel optimization, where sequences are iteratively constructed and improved by adaptive priority rules. A series of new crossover operators for GAs is presented in [20], showing good results on instances of the CSPLib. These results are further improved by applying a subsequent local search. Furthermore, ant colony optimizations are presented in [11], [5], [21], [22] applying different pheromone trails and transition rules. In summary, local search procedures applying a set of neighborhood operators show the overall best results on CS instances.

Few authors provide hybrid algorithms for CS, combining exact and heuristic approaches. In [8] an ILP is incorporated into a variable local search procedure producing competitive results for the industrial version of CS. In [23] the authors previous GA approach is extended by incorporating ILP formulations into the crossover operator. The solution quality of

the resulting GA on the CSPLib instances is increased, but at the cost of a very high runtime.

III. Fitness Landscape Analysis of Car Sequencing

In general, a fitness landscape $(\mathcal{X}, f, d)$ of an instance of a combinatorial optimization problem consists of a set of solutions $\mathcal{X}$, a fitness function (objective function) $f : \mathcal{X} \to \mathbb{R}$, which assigns an objective value to each of the solutions in $\mathcal{X}$, and a distance measure d, which defines the spatial structure of the landscape. For the notion of distances a so-called neighborhood operator $\mathcal{N}$ is applied. $\mathcal{N}$ converts a solution $x \in \mathcal{X}$ into a new solution $x' \in \mathcal{N}(x) \subseteq \mathcal{X}$ by changing the composition of x. With the neighborhood operator $\mathcal{N}$, the search space can be interpreted as an undirected graph $\mathcal{G}_{\mathcal{N}} = (\mathcal{X}, E)$ with $\mathcal{X}$ being the vertices of the graph and edge set $E = \{(x, x') \in \mathcal{X} \times \mathcal{X} \mid x' \in \mathcal{N}(x)\}$ introducing an edge between x and x', if x' can be reached from x by one application of $\mathcal{N}$ or vice versa. The distance $d(x, y)$ between two solution x and y is then defined as the length of the shortest path from x to y in $\mathcal{G}_{\mathcal{N}}$, which equals the minimum number of applications of $\mathcal{N}$ to transform x into y or vice versa.

For CS, we analyze the fitness landscapes for a set of nine widely applied problem instances in the literature. The instances are available in the CSPLib, an online library for constraint satisfaction problems. Each instance has $T = 100$ production slots, $M =$ 19-26 models with $O = 5$ options to be sequenced. For every instance, we generate different fitness landscapes using equation (1) as fitness function and four neighborhood operators found in the literature [13]. We study the autocorrelation of the resulting landscapes in terms of a random walk analysis and perform a fitness-distance correlation analysis.

A. Representation and Neighborhood Operators

A solution for CS is represented by a permutation with repetitions. Thus, a sequence is encoded as a vector of length T, where a value $m \in M$ at position $t = 1, \ldots, T$ indicates that a copy of model m is produced at the tth production slot. We only consider feasible solutions, where the sum of occurrences of each model $m \in M$ in the sequence corresponds to the models demand d_m. This is ensured by an appropriate initialization of solutions and neighborhood operators that maintain feasibility.

For the fitness landscape analysis of CS, we consider four different neighborhood operators $\mathcal{N}$ [13], that are currently applied in the literature. The operators are along the line of operators for the traveling salesman problem:

- Swap neighborhood $\mathcal{N}_{\text{sw}}$: Two models in the sequences exchange their positions.
- Adjacent swap neighborhood $\mathcal{N}_{\text{ad}}$: Two adjacent models in the sequences exchange their positions.
- Shift neighborhood $\mathcal{N}_{\text{sh}}$: A model is forward or backward shifted a certain number of positions in the sequence.
- Reverse neighborhood $\mathcal{N}_{\text{re}}$: The order of a subsequence of models is reversed.

B. Autocorrelation

The autocorrelation analysis studies the ruggedness of a landscape, which is important for local search procedures. A fitness landscape is said to be rugged, if no correlation between the distance of solutions and their fitness values exists. Thus, a small distance between two solutions can imply a large difference in their objective values. Rugged landscapes are difficult to search for guided local search methods, which traverse the search space by iteratively sampling new solutions in the neighborhood of a current solution. Thereby, the fitness value of the current solution is employed to decide whether the search proceeds with one neighboring solution or not. The search is guided from low quality solutions to higher quality solutions in order to find the global optimum. However, on a rugged landscape, the fitness values of neighboring solutions are not correlated and can, therefore, not be used to guide the search process, which results in a merely random search. In contrast, if a correlation between the distance and the fitness value of solutions exists, the respective landscape is said to be smooth and is adequate for guided local search methods.

In [24] the autocorrelation function is introduced to compute the ruggedness of a landscape. The approach requires to evaluate the entire search space of a problem instance, thus, all solutions in $\mathcal{X}$. Since for many optimization problems, including CS, the number of solutions increases exponentially depending on some input factor, Weinberger [25] proposes to use a random walk to estimate the autocorrelation. Beginning with an arbitrary solution, a random walk picks a random solution in the neighborhood of the current solution and proceeds the walk with the new solution. This move is repeated until a maximum number m of walking steps is reached. The fitness values $f(x)$ of all solutions x visited during the walk are used to compute the random walk correlation function $r(s)$ as follows

$$r(s) = \frac{1}{\sigma^2(f)(m-s)} \sum_{t=1}^{m-s} (f(x_t) - \overline{f})(f(x_{t+s}) - \overline{f}) \quad (7)$$

With $\sigma^2(f)$ being the variance of the fitness values, $r(s)$ computes the correlation of all solutions that are s steps away along the random walk of length m. Based on the nearest-neighbor correlation of the landscape $r(1)$, which is the correlation of neighboring solutions, the correlation length l of the landscape is defined as [26]

$$l = \begin{cases} 0, & \text{if } r(1) = 0 \\ -\frac{1}{\ln(|r(1)|)}, & \text{if } r(1) \neq 0 \end{cases} \quad (8)$$

l reflects the ruggedness of the landscape. The lower the correlation length the more rugged the landscape. Since the correlation length depends on the applied neighborhood operator as well as on the size of the instance, different instances and/or neighborhood operators should be compared by the normalized correlation length l' [27, p. 229]

$$l' = \frac{l}{diam(\mathcal{G}_{\mathcal{N}})} \quad (9)$$

Table II
DIAMETERS OF NEIGHBORHOOD OPERATORS

$\mathcal{N}_{\text{sw}}$	$T-1$
$\mathcal{N}_{\text{ad}}$	$\frac{T(T-1)}{2}$
$\mathcal{N}_{\text{sh}}$	$T-1$
$\mathcal{N}_{\text{re}}$	$T-1$

with $diam(\mathcal{G}_{\mathcal{N}})$ being the diameter of the search space, which is the maximum distance of any two solutions in $\mathcal{G}_{\mathcal{N}}$. The diameters of the considered neighborhood operators are listed in Table II.

For each CS instance, we get four different fitness landscapes by applying the neighborhood operators of Section III-A together with equation (1) as fitness function. To measure the autocorrelation of the resulting landscapes, we perform a random walk with 100,000 steps on each landscape and determine the random walk correlation coefficient of neighboring solutions $r(1)$ and the normalized correlation length l'. Table III shows the respective results.

The results of the correlation coefficients $r(1)$ suggest a smooth landscape for all instances and neighborhood operators, especially for $\mathcal{N}_{\text{ad}}$. However, since $\mathcal{N}_{\text{ad}}$ has a larger diameter as compared to the other operators, the normalized correlation lengths l', suggests $\mathcal{N}_{\text{sh}}, \mathcal{N}_{\text{re}}$ and $\mathcal{N}_{\text{sw}}$ to be favorable for local search procedures.

C. Fitness-Distance-Correlation

The Fitness-Distance-Correlation (FDC) [3] is a measure for problem difficulty for evolutionary algorithms. The FDC measures the correlation between the fitness differences of a solution and the global best solution and their distances in the search space. Thus, with a sample of solutions, the FDC coefficient ϱ is defined as

$$\varrho(f, d_{\text{opt}}) = \frac{\text{cov}(f, d_{\text{opt}})}{\sigma(f)\sigma(d_{\text{opt}})} \tag{10}$$

with d_{opt} being a solutions distance to the nearest optimal solution and $\text{cov}(f, d_{\text{opt}})$ the covariance of f and d_{opt}. A value of $\varrho = 1$ indicates a perfect correlation between fitness and distance to the optimum. The more both values are correlated, the easier the resulting problem is for selection-based algorithms as a path of solutions with increasing fitness values leads to the optimum [24]. The problem difficulty can be classified [3] according to ϱ. A value of $\varrho \geq 0.15$ suggests a straightforward minimization problem for evolutionary algorithms, while lower values of ϱ indicate an uncorrelated or even misleading landscape. The FDC coefficient can be computed for random solutions as well as locally optimal solutions. The usage of locally optimal solutions can give further insights into the global structure of the search space [24]. Additionally, fitness-distance plots are suitable for the interpretation of the results.

To compute the FDC coefficient, the distances between solutions have to be known. The actual distance between two solutions is the minimum number of a specific neighborhood operation in order to transform one solution into another. However, the computation of the distances is not straightforward for every neighborhood operator in Section III-A. For instance, no polynomial algorithm is available to compute the distances of the $\mathcal{N}_{\text{re}}$ operator [28]. Thus, in order to allow a comparison of the resulting FDC coefficients for different neighborhood operators, we apply the following surrogate distance metrics [29]:

- Adjacency distance metric d_{adj}: The bidirectional adjacency distance computes how often a pair of models is adjacent in both sequences
- Precedence distance metric d_{prec}: The precedence distance computes how often a model m is preceded by a model m' in both sequences.
- Absolute position distance metric d_{abs}: The absolute position distance computes the number of times the position of models is identical in both sequences.

The combination of the four neighborhood operators with the three distance approximations leads to twelve fitness landscapes overall for each instance. For each neighborhood operator, we determine 1000 local optima by applying a steepest ascent hill climbing algorithm using the respective operator which starts from a random solution and stops if a local optimum is reached. The global optimum for each instance is known, except for instance 19-71, where we use the best known solution instead. The global optima are obtained by an iterative beam search (IBS) algorithm [10]. The FDC coefficients are calculated for the distance to the global optimum (ϱ_{global}) as well as for the average distances to all other local optima (ϱ_{local}). Table IV presents the respective results.

The results suggest that the adjacency distance d_{adj} is the best metric to describe structural properties of local optimal solutions, as the resulting correlation coefficients are higher compared to the other metrics. Thus, a certain amount of adjacent model pairs are shared by high quality solutions. According to [3], $\varrho_{global} \geq 0.15$ and $\varrho_{local} \geq 0.15$ indicate that the resulting landscapes for all instances and neighborhood operators are suitable for EAs using the adjacency distance metric. A value of $\varrho_{global} \geq 0.15$ means, that the fitness and the distance to the global best solution are positively correlated since the smaller the distance to the global best solution the lower the fitness value of a solution. Note, that in our case solutions with a low fitness are favored, since CS is a minimization problem. $\varrho_{local} \geq 0.15$ indicates that the lower the fitness value of a solution the smaller the average distance to all other local optima. Thus, high quality local optima lie in the center of other local optima.

In Figure 1, we show some scatter plots for a representative example using instance 10-93 and $\mathcal{N}_{\text{re}}$. Figures 1(a) and 1(b) plot for each local optimum its adjacency distance to the global optimum and its average adjacency distance to all other local optima, respectively, against its absolute fitness deviation to the global optimum. We can observe the aforementioned positive correlation between the adjacency distance and the fitness of a solution. Furthermore, all local optima are clustered in a small region of the search space as the maximum

Table III
RANDOM WALK CORRELATION COEFFICIENTS AND NORMALIZED CORRELATION LENGTHS

instance	r(1)				l'			
	$\mathcal{N}_{sw}$	$\mathcal{N}_{ad}$	$\mathcal{N}_{sh}$	$\mathcal{N}_{re}$	$\mathcal{N}_{sw}$	$\mathcal{N}_{ad}$	$\mathcal{N}_{sh}$	$\mathcal{N}_{re}$
4-72	0.9512	0.9880	0.9648	0.9621	0.2019	0.0167	0.2817	0.2614
6-76	0.9543	0.9878	0.9671	0.9646	0.2162	0.0165	0.3015	0.2805
10-93	0.9526	0.9877	0.9654	0.9631	0.2082	0.0163	0.2867	0.2683
16-81	0.9516	0.9876	0.9642	0.9621	0.2038	0.0162	0.2775	0.2613
19-71	0.9533	0.9880	0.9654	0.9637	0.2111	0.0168	0.2867	0.2732
21-90	0.9550	0.9879	0.9673	0.9648	0.2195	0.0166	0.3040	0.2821
26-92	0.9540	0.9886	0.9668	0.9641	0.2147	0.0176	0.2994	0.2765
41-66	0.9540	0.9887	0.9677	0.9648	0.2143	0.0178	0.3076	0.2820
26-82	0.9537	0.9885	0.9658	0.9642	0.2130	0.0175	0.2902	0.2767

Table IV
FITNESS-DISTANCE CORRELATION COEFFICIENTS ϱ

instance		$\mathcal{N}_{sw}$			$\mathcal{N}_{ad}$			$\mathcal{N}_{sh}$			$\mathcal{N}_{re}$		
		d_{adj}	d_{prec}	d_{abs}	d_{adj}	d_{prec}	d_{abs}	d_{adj}	d_{prec}	d_{abs}	d_{adj}	d_{prec}	d_{abs}
4-72	ϱ_{global}	0.385	0.041	0.110	0.419	0.302	0.023	0.404	0.001	0.015	0.504	0.106	0.117
	ϱ_{local}	0.343	0.025	0.194	0.319	-0.085	-0.416	0.451	0.074	-0.069	0.510	0.127	0.245
6-76	ϱ_{global}	0.256	-0.024	0.015	0.388	0.054	-0.109	0.283	0.006	-0.062	0.319	0.048	0.102
	ϱ_{local}	0.365	0.032	0.076	0.284	-0.077	-0.387	0.429	0.073	-0.019	0.466	0.031	0.126
10-93	ϱ_{global}	0.260	-0.062	-0.005	0.575	0.276	-0.012	0.350	-0.039	-0.015	0.504	-0.001	0.063
	ϱ_{local}	0.340	-0.027	0.009	0.237	-0.146	-0.441	0.436	-0.065	-0.112	0.579	0.019	0.094
16-81	ϱ_{global}	0.300	0.056	0.044	0.451	0.256	0.077	0.357	0.061	0.060	0.470	0.102	0.103
	ϱ_{local}	0.425	-0.038	0.136	0.288	-0.054	-0.264	0.419	0.020	-0.026	0.557	0.070	0.158
19-71	ϱ_{global}	0.202	0.030	0.045	0.565	0.282	-0.061	0.317	0.036	0.039	0.336	0.082	0.042
	ϱ_{local}	0.354	0.026	0.076	0.165	-0.156	-0.507	0.451	-0.035	-0.140	0.541	0.112	0.148
21-90	ϱ_{global}	0.306	-0.010	0.043	0.541	0.286	0.003	0.309	0.006	0.021	0.364	0.045	0.063
	ϱ_{local}	0.375	0.038	0.062	0.268	-0.067	-0.372	0.469	-0.001	-0.034	0.573	-0.004	0.158
36-92	ϱ_{global}	0.191	0.068	0.041	0.610	0.138	-0.168	0.317	0.006	-0.039	0.384	0.022	0.071
	ϱ_{local}	0.359	0.034	0.111	0.135	-0.139	-0.476	0.465	0.011	-0.121	0.559	0.114	0.208
41-66	ϱ_{global}	0.112	0.024	0.002	0.427	0.048	-0.062	0.187	-0.189	-0.077	0.199	0.030	0.081
	ϱ_{local}	0.256	0.044	0.105	0.222	-0.175	-0.368	0.406	-0.086	-0.187	0.384	0.004	0.102
26-82	ϱ_{global}	0.289	0.010	0.090	0.525	0.316	-0.015	0.322	-0.068	0.025	0.403	0.088	0.024
	ϱ_{local}	0.367	0.052	0.194	0.242	-0.108	-0.512	0.413	-0.022	0.041	0.503	0.085	0.164

distance between any two local optima is 12,15 (see Figure 1(b)). Figures 1(c) and 1(d) present the scatter plots for the precedence distance and absolute distance, respectively, of each solution to the global optimum against its absolute fitness deviation to the global optimum. Both distance metrics seem to have a low correlation which confirms the results of Table IV. Again, for d_{prec}, the local optima appear to be grouped in a small region around the global optimum, whereas, for d_{abs}, the distances to the global optimum are close to the diameter of the search space which suggests that the local optima are evenly distributed in the landscape. Interpreting the plots, we assume the existence of a 'big valley' structure for d_{adj}, since local optima are accumulated in a small region of the search space with high quality solutions being in the center of the local optima.

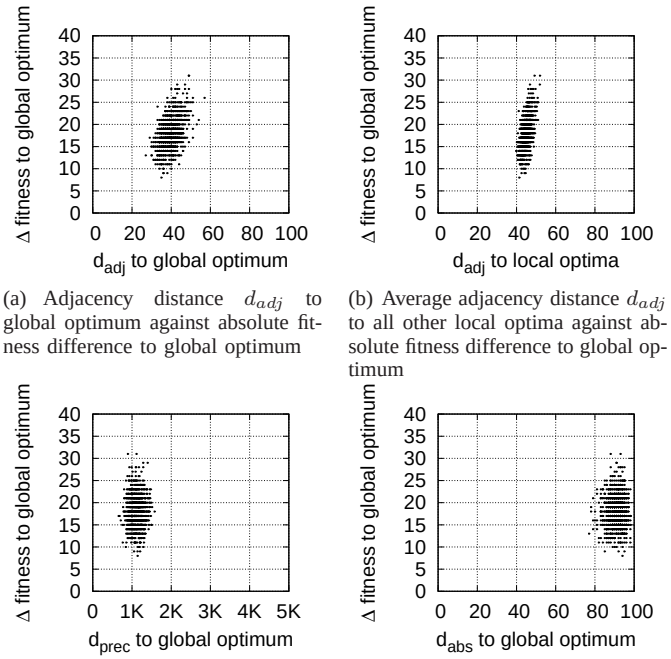
IV. METAHEURISTICS FOR CAR SEQUENCING

In this section, we describe two metaheuristics for CS, a variable neighborhood search (VNS) and a memetic algorithm (MA), whose designs are based on the results of the preceding landscape analysis.

A. Variable Neighborhood Search

Local search (LS) algorithms explore the fitness landscape of a problem instance by iteratively moving from one solution to a new neighboring solution with a better fitness, until a local optimum is reached. In order to avoid being trapped in the first local optimum found, several metaheuristics based on LS have been developed, like simulated annealing [30], which accepts worse solutions with a certain probability, or tabu search [31], where the last visited solutions are stored in a tabu list so that they are not revisited repeatedly. A different metaheuristic, called variable neighborhood search (VNS) was proposed by [32], where instead of merely one neighborhood operator, a set of neighborhoods is applied. Thus, VNS can escape a local optimum in one landscape, by using a different neighborhood operator and, thus, changing the landscape. The original scheme of VNS arranges all neighborhood operators in a certain sequence with increasing neighborhood size. If VNS experiences a local optimum using one neighborhood operator, the search proceeds with the next operator in the list until a certain stopping criterion is reached.

In order to decide if the current solution is a local op-



(a) Adjacency distance d_{adj} to global optimum against absolute fitness difference to global optimum

(b) Average adjacency distance d_{adj} to all other local optima against absolute fitness difference to global optimum

(c) Precedence distance d_{prec} to global optimum against absolute fitness difference to global optimum

(d) Absolute distance d_{abs} to global optimum against absolute fitness difference to global optimum

Figure 1. Scatter plots for instance 10-93 and $\mathcal{N}_{\text{re}}$ based on 1000 local optima.

Input: initial solution x
Output: best solution found
  Initialize probabilities ρ
  Evaluate(x)
  for i := 1 to N_{VNS} **do**
    $\mathcal{N} \leftarrow$ SelectNeighborhoodOperator(ρ)
    $x' \leftarrow \mathcal{N}(x)$
    Evaluate(x')
    if $x'.fitness \leq x.fitness$ **then**
      $x \leftarrow x'$
    end if
    $i \leftarrow i + 1$
  end for

Figure 2. Outline of VNS algorithm

timum, a systematic exploration of its entire neighborhood is required. Since for large neighborhoods this can be computational demanding, we use a different approach of VNS for the CS problem, inspired by [13]. Instead of applying the neighborhood operators in a predefined sequence, we choose a neighborhood operator with a certain probability p at each step of the search. The operator is then applied to the current solution and the search proceeds with the new found neighboring solution if its fitness value is not worse compared to the current solution. Allowing the search to continue with solutions having the same fitness value as the current solution is useful to traverse plateaus of equal fitness in the landscape. The search is stopped after a maximum number of steps N_{VNS} is reached. The algorithm is outlined in Figure 2.

The probabilities p of the different neighborhood operators

Table V
DISTANCES BETWEEN REDUNDANT CS SOLUTIONS

$\mathcal{N}_{\text{swa}}$	$\begin{cases} \frac{T}{2} & \text{, for even } T \\ \frac{T-1}{2} & \text{, for odd } T \end{cases}$
$\mathcal{N}_{\text{adj}}$	$\frac{T(T-1)}{2}$
$\mathcal{N}_{\text{ins}}$	$T-1$
$\mathcal{N}_{\text{rev}}$	1

are obtained using the results of Section III-B, where the autocorrelation of the landscape induced by each operator is analyzed. It is assumed that the smoother the landscape, the better the performance of LS [27]. Regarding the normalized correlation lengths l' in Table III, the shift operator $\mathcal{N}_{\text{sh}}$ leads to the smoothest landscape, followed by $\mathcal{N}_{\text{re}}$ and $\mathcal{N}_{\text{sw}}$. However, we also have to consider the locality induced by our representation and neighborhood operators [33], [34]. Locality describes how well the genotypic neighborhood corresponds to the phenotypic neighborhood of a solution [35]. Our representation of solutions leads to redundant solutions in the genotypic search space, as for each sequence a symmetric solution with equal fitness can be found by inverting the sequence. A neighborhood operator which induces a large distance between both, the sequence and its inverted counterpart, results in a low locality, as two actually identical areas in the genotypic landscape are far away from each other. Thus, high quality solutions are also clustered in two regions of the search space, from which merely one is explored by a LS algorithm. In contrast, LS benefits from a small distance between both redundant solutions as the search can be intensified in one small area of the search space where high quality solutions are concentrated. For each neighborhood operator, the distance between a sequence and its inverted counterpart is shown in Table V.

For the $\mathcal{N}_{\text{ad}}$ and $\mathcal{N}_{\text{sh}}$ operator, the distance between two redundant solutions equals the diameter of the respective landscapes (compare with Table II), thus, both solutions are maximally away from each other in the search space. For the $\mathcal{N}_{\text{sw}}$ operator, the distance between two redundant sequences is half the diameter of its landscape. Using $\mathcal{N}_{\text{re}}$, the inverted solution can be reached in one operation as both solutions are neighbors. In summary, the $\mathcal{N}_{\text{re}}$ neighborhood operator seems most promising for LS, since it results in a smooth landscape and induces a high locality of redundant solutions. Followed by $\mathcal{N}_{\text{sw}}$ and $\mathcal{N}_{\text{sh}}$, where $\mathcal{N}_{\text{sh}}$ leads to a smoother landscape compared to $\mathcal{N}_{\text{sw}}$, but $\mathcal{N}_{\text{sw}}$ results in a higher locality of redundant solutions. Thus, we base our VNS on these three operators and assign $\mathcal{N}_{\text{re}}$ a probability of 60% to be chosen at each step and $\mathcal{N}_{\text{sw}}$ and $\mathcal{N}_{\text{sh}}$ each a probability of 20%. We ignore the $\mathcal{N}_{\text{ad}}$ operator since its normalized correlation length l' is low and it induces a low locality of redundant solutions.

B. Memetic Algorithm

Memetic algorithms (MA) [36] are a conjunction of evolutionary algorithms (EA) and local search (LS). Thus, they combine the concept of population-based evolution with individual learning. In the literature, MAs are also referred to

```
Output: best solution found
  InitialPopulation P ← IBS(ω)
  Evaluate(P)
  Sort(P)
  for i := 1 to N_MA do
    Parents p1, p2 ← TournamentSelection(size = 2)
    Offspring o1, o2 ← Crossover(p1, p2)
    Evaluate(o1, o2)
    if o1.fitness ≤ o2.fitness then
      j ← 1
    else
      j ← 2
    end if
    oj ← VNS(oj)
    x ← oj with x being the last element in P
    for all x ∈ P do
      if RandomNumber(0,1) ≤ pm then
        x ← Mutate(x)
      end if
    end for
    for all x ∈ P do
      if RandomNumber(0,1) ≤ pls then
        x ← VNS(x)
      end if
    end for
    Sort(P)
    i ← i + 1
  end for
```

Figure 3. Outline of MA

as hybrid genetic algorithms or genetic local search. They are similar to EAs in that a population of individual solutions to a problem is iteratively modified by evolutionary recombination and mutation operators, in order to explore the search space and guide the search to promising areas. In contrast to EAs, an additional local search is performed at each generation, which locally improves individuals of the population to intensify the search in promising areas. MAs have been successfully applied to other scheduling problems like parallel-machine scheduling, job-shop scheduling or flow-shop scheduling [37], [38]. Comprehensive introductions to MAs can be found in [39] and [40]. Algorithm 3 outlines the general structure of our MA for the CS problem, which is described in detail in the following.

A population P in EAs consists of a set of individual solutions. The size of the population $|P|$ in MAs is usually much smaller than in traditional EAs due to the complexity of the local search, which inhibits the evolution of large populations [24]. The initial population can be set up randomly or obtained by a heuristic procedure. Using heuristics for the initialization phase usually improves the performance of an EA as fewer generations are needed to guide the population to promising areas in the search space. Thus, we initialize our population using a heuristic iterative beam search (IBS) algorithm for CS [10] with beam width ω.

The population-based evolution in MAs is achieved by selection, recombination and mutation. At each generation of our MA approach, two solutions, named parents, are selected for crossover by a binary tournament selection. The crossover operator combines parts of both solutions in order to derive two offspring solutions. This corresponds to the reproduction in biological evolution. In EAs, the crossover operator is used to intensify the search in promising areas of the landscape, thus, existing similarities of parent solutions should be preserved in the offspring. This characteristic is called respectfulness [41] and is necessary for successful evolutionary search [42]. Especially in the presence of a big valley, respectful crossover is likely to perform well, as high quality solutions can be found in the vicinity of other good solutions. The preceding FDC analysis of section III-C suggests that the adjacency distance metric is suited to describe similarities between high quality solutions for CS and even leads to a big valley. Therefore, the crossover operator should preserve adjacent relationships of models existing in both parents.

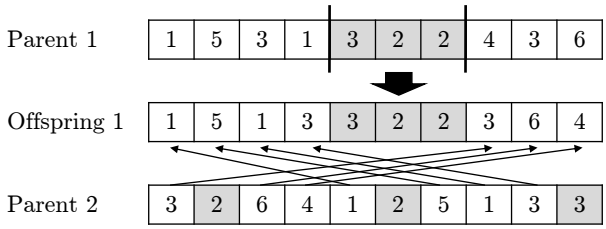


Figure 4. Order Crossover (OX)

We consider three recombination operators for CS, Order Crossover (OX) [43] and propose a heuristic variant of OX (hOX), as well as Non Conflict Position Crossover (NCPX), which showed good results in previous experiments [20]. OX, outlined in Figure 4, randomly selects a subsequence of one parent and transfers it to the corresponding slots of one offspring. Beginning at the second crossover site of the offspring, the remaining models are allocated according to their occurrence in the second parent. OX preserves the absolute positions of a subsequence in the first parent and the relative order of models in the alternative parent. It is called a blind recombination operator as it uses no problem-specific knowledge. Additionally, we propose a heuristic variant of OX (hOX), where we choose the subsequence of parent 1 such that it is delimited by violated slots. A slot is violated, if the respective model at this slot induces at least one sequencing rule violation. If a sequence contains only one violated slot, the second crossover site is chosen randomly. The remaining models are assigned from the second parent the same way as in OX, but starting at the first slot of the offspring. To apply hOX, a vector of length T indicating the violated slots has to be stored and updated during the search. The third recombination operator NCPX is also a heuristic operator and was proposed for CS in [20]. It is outlined for an example sequence in Figure 5. First a random number nb_g between 0 and the number of non-violated slots is chosen. Our example has 7 non-violated slots and nb_g amounts to 5. Then, a random starting point Pos_d is selected from which nb_g models at non-

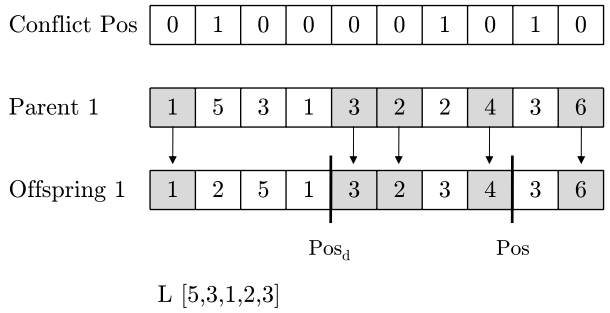


Figure 5. Non Conflict Position Crossover (NCPX)

violated slots are copied from parent one to the corresponding slots at the offspring. If the end of the sequence is meanwhile reached, the copy process proceeds at the beginning of the sequence. After nb_g models are copied, the remaining models form a list of interest L. Another random position Pos is chosen, from which the models in L are assigned to the empty slots according to a heuristic function which considers the induced number of violations by a model as well as a so-called utilization rate [11]. If models are tied in the resulting number of violations as well as in the utilization rate and one of these model occupies the same slot in the second parent, this model is selected. Alternatively, ties are broken randomly. Note, that only in case of ties a second parent is considered by NCPX and an offspring is created by merely a single parent, otherwise. NCPX preserves absolute positions of nb_g models of the parent solution.

We examine the resulting similarity of parents and offspring according to the adjacency distance metric d_{adj} for the different crossover operators. Therefore, 1000 recombinations are performed with randomly selected parents for each crossover operator and the adjacency distance between both parents is compared with the resulting average adjacency distance between both offspring and parents. Figure 6 plots the results for the representative instance 4-72. We can observe, that OX and hOX produce offspring with a lower average distance to both parents compared to NCPX. For both operators, the distance between offspring and parents becomes smaller with decreasing distance between both parents. Thus, offspring are likely to be produced in the vicinity of both parent solutions. In contrast, NCPX doesn't seem to preserve existing adjacency relations as the average distances of the offspring are independent of the distance between both parents.

After recombination, the offspring with the least number of violations is locally improved using the VNS algorithm of Section IV-A. VNS introduces individual learning to the MA. The resulting solution replaces the current worst solution in the population P. Thus, we perform a so-called steady-state selection scheme [44], where one individual in the population is replaced at each generation.

The mutation operator is used for diversification in population-based EAs. It randomly changes individuals in the population in order to explore new regions in the search space and prevent an early convergence to a single local optimum. In our MA, individuals are mutated by a single application of $\mathcal{N}_{\text{re}}$, thus, a random subsequence of a solution is reversed. We perform mutation with probability p_m after recombination and the local improvement of the best offspring. After mutation, each generation concludes with another application of VNS. The VNS is applied to each individual in the population depending on a certain probability p_{ls}.

The MA is stopped after reaching the maximum number of generations N_{MA}.

V. Experiments

A. Experimental Setup

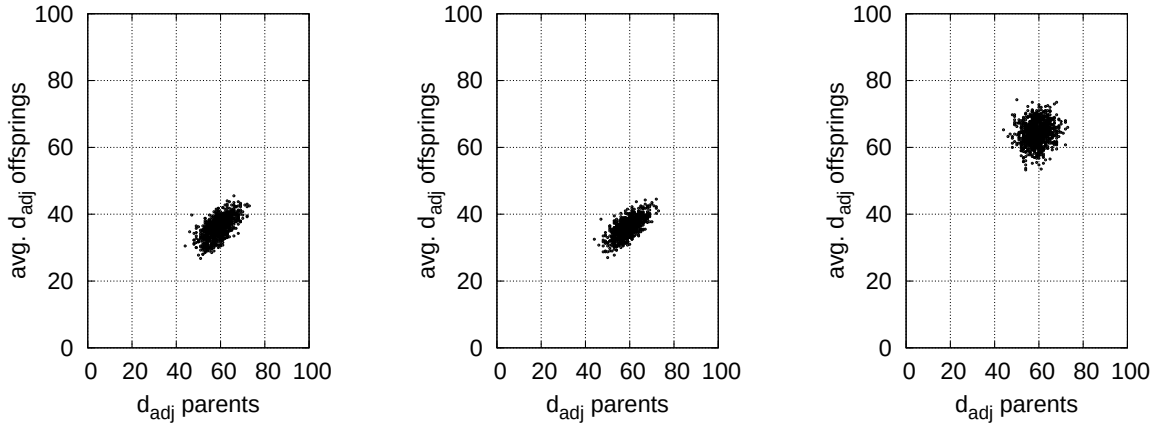
We evaluate the proposed algorithms on the problem instances for CS available in the CSPLib, an online library of constraint satisfaction problems. The instances are divided in two sets. The first set consists of the aforementioned nine instances, all with a sequence length $T = 100$, 5 options and 19-26 models. The second set is composed of 30 larger problem instances with 200-400 production slots, 5 options and 19-26 models.

The algorithms are implemented in JAVA. All experiments run on an Intel Xeon X5570 with 2.93GHz using 4GB RAM.

B. Results

The algorithms are applied with parameters from Table VI. We perform two versions of VNS with different neighborhood operators and probabilities for selecting neighborhoods. VNS_1 corresponds to Section IV-A using a set of three neighborhood operators with probabilities $\{\mathcal{N}_{\text{re}}, \mathcal{N}_{\text{sw}}, \mathcal{N}_{\text{sh}}\} = \{0.6, 0.2, 0.2\}$. VNS_2 applies all neighborhood operators of Section III-A with equal probabilities, thus, $\{\mathcal{N}_{\text{sw}}, \mathcal{N}_{\text{ad}}, \mathcal{N}_{\text{sh}}, \mathcal{N}_{\text{re}}\} = \{0.25, 0.25, 0.25, 0.25\}$. The initial solution for both VNS' is obtained by the aforementioned IBS algorithm with beam width $\omega = 5$. Starting from the initial solution, VNS_1 and VNS_2 are applied with a maximum of 50,000*T moves, where T is the number of production slots in the sequence and the best sequence found is returned.

Furthermore, we consider three MA algorithms each with a different crossover operator. Thus, MA_{OX} uses the Order Crossover, MA_{hOX} the heuristic Order Crossover and MA_{NCPX} the Non Conflict Position Crossover. All MAs have a population size $|P|$ of 20 individuals and the initial population is set up with the best 20 individuals found by the IBS algorithm with $\omega = 100$. Within the MAs, VNS_1 is applied with at most 500*T moves. The mutation and local search probabilities are set to 0.05 and 0.1, respectively. Table VII shows the performance results of the five considered algorithms on both sets of problem instances. The results are obtained by 10 independent runs of each algorithm on each instance. obj* states the currently best known solutions as obtained by a genetic algorithm with a subsequent local search [20]. For each algorithm and instance, we present the number of sequencing rules of the best solution found (column 'best obj.'), the resulting average number of sequencing rules (column 'avg. obj.), as well as the average time (column avg. time) in seconds required for the 10 runs. To exemplify the results, the best average objective values are highlighted in gray and new best solutions found are marked with an asterisk.



(a) Order Crossover (OX)

(b) Heuristic Order Crossover (OX^h)

(c) Non Conflict Position Crossover (NCPX)

Figure 6. Adjacency distance d_{adj} between random parents against the average adjacency distance d_{adj} between the resulting offspring and their parents for different crossover operators (based on instance 4-72).

Table VII
RESULTS ON PROBLEM SETS 1 AND 2

instance	obj*	VNS_1			VNS_2			MA_{OX}			MA_{hOX}			MA_{NCPX}		
		best obj.	avg. obj.	avg. time[s]	best obj.	avg. obj.	avg. time[s]	best obj.	avg. obj.	avg. time[s]	best obj.	avg. obj.	avg. time[s]	best obj.	avg. obj.	avg. time[s]
4-72	0	0	0.0	0.61	0	0.0	1.28	0	0.0	1.21	0	0.0	1.07	0	0.0	1.10
6-76	6	6	6.0	13.01	6	6.0	12.49	6	6.0	41.11	6	6.0	67.16	6	6.0	67.87
10-93	3	3	3.0	12.98	3	3.0	12.76	3	3.0	41.51	3	3.0	69.13	3	3.0	69.33
16-81	0	0	0.0	2.69	0	0.0	4.17	0	0.0	1.43	0	0.0	1.44	0	0.0	1.46
19-71	2	2	2.0	12.91	2	2.0	12.78	2	2.0	41.03	2	2.0	62.09	2	2.0	62.59
21-90	2	2	2.0	12.85	2	2.0	12.77	2	2.0	41.30	2	2.0	61.58	2	2.0	62.05
36-92	2	2	2.0	12.87	2	2.0	12.76	2	2.0	41.18	2	2.0	66.38	2	2.0	66.30
41-66	0	0	0.0	0.10	0	0.0	0.09	0	0.0	0.73	0	0.0	0.76	0	0.0	0.80
26-82	0	0	0.0	0.66	0	0.0	1.74	0	0.0	1.10	0	0.0	1.11	0	0.0	1.14
200-01	0	0	0.0	10.17	0	0.1	11.17	0	0.1	47.25	0	0.1	39.17	0	0.1	65.43
200-02	2	2	2.0	31.88	2	2.0	30.53	2	2.0	96.22	2	2.0	167.65	2	2.0	168.97
200-03	4	3*	4.4	32.03	4	4.7	30.72	4	5.1	96.41	4	4.7	188.92	4	5.0	193.63
200-04	7	7	7.0	32.06	7	7.0	30.42	7	7.0	95.96	7	7.0	191.35	7	7.0	193.58
200-05	6	6	6.0	31.74	6	6.0	30.59	6	6.0	95.82	6	6.0	178.32	6	6.0	182.28
200-06	6	6	6.0	31.37	6	6.0	30.66	6	6.0	95.78	6	6.0	179.57	6	6.0	180.31
200-07	0	0	0.0	0.29	0	0.0	0.45	0	0.0	2.54	0	0.0	2.64	0	0.0	2.57
200-08	8	8	8.0	31.11	8	8.0	30.77	8	8.0	94.66	8	8.0	190.78	8	8.0	191.45
200-09	10	10	10.0	31.27	10	10.0	30.70	10	10.0	95.89	10	10.0	199.32	10	10.0	198.95
200-10	19	19	19.0	31.12	19	19.1	30.57	19	19.0	94.62	19	19.0	223.98	19	19.0	224.70
300-01	0	0	0.0	18.82	0	0.4	39.84	0	0.6	121.15	0	0.4	207.64	0	0.4	212.76
300-02	12	12	12.0	54.75	12	12.0	52.25	12	12.0	168.08	12	12.0	351.28	12	12.0	350.90
300-03	13	13	13.0	53.61	13	13.0	52.30	13	13.0	163.45	13	13.0	351.87	13	13.0	356.48
300-04	7	7	7.2	53.83	7	7.4	52.17	7	7.1	163.74	7	7.1	329.10	7	7.1	340.02
300-05	29	27*	29.2	54.72	29	29.9	52.02	29	29.7	162.74	29	29.7	398.45	29	29.8	404.93
300-06	2	2	2.0	55.63	2	2.2	51.92	2	3.1	164.27	2	3.2	308.74	3	3.3	312.04
300-07	0	0	0.0	3.63	0	0.0	6.80	0	0.0	15.16	0	0.0	19.55	0	0.0	20.63
300-08	8	8	8.0	53.87	8	8.0	52.02	8	8.0	163.97	8	8.0	333.13	8	8.0	336.66
300-09	7	7	7.0	54.96	7	7.3	52.09	7	7.0	162.38	7	7.0	345.58	7	7.0	345.86
300-10	21	21	21.0	55.27	21	21.1	52.11	21	21.0	159.86	21	21.0	390.76	21	21.0	400.17
400-01	1	1	1.1	83.29	1	1.3	77.44	1	1.8	242.35	1	1.7	432.35	1	1.9	435.24
400-02	15	15	15.4	83.19	15	15.6	78.04	16	16.2	238.09	15	15.8	530.02	16	16.4	535.58
400-03	9	9	9.1	83.38	9	9.2	77.23	9	9.0	243.58	9	9.0	507.20	9	9.0	517.08
400-04	19	19	19.0	82.80	19	19.0	77.64	19	19.0	243.77	19	19.0	564.87	19	19.0	566.98
400-05	0	0	0.0	0.25	0	0.0	0.20	0	0.0	5.33	0	0.0	11.73	0	0.0	4.44
400-06	0	0	0.0	6.01	0	0.0	7.76	0	0.0	18.31	0	0.0	38.60	0	0.0	38.60
400-07	4	4	4.0	82.43	4	4.0	77.80	4	4.3	244.32	4	4.0	470.11	4	4.1	474.97
400-08	4	4	4.0	81.69	4	4.0	77.84	4	4.0	243.84	4	4.0	466.17	4	4.0	468.50
400-09	5	5	6.6	81.64	6	6.9	78.09	6	6.8	242.96	6	6.9	519.20	6	6.9	524.94
400-10	0	0	0.0	6.55	0	0.0	11.98	0	0.0	7.04	0	0.0	11.68	0	0.0	10.79

Table VI
PARAMETERS OF ALGORITHMS IN THE EXPERIMENTS

Parameter		VNS	MA
IBS beam width	ω	5	100
Number of MA generations	N_{MA}	-	100
Number of VNS moves	N_{VNS}	50,000*T	500*T
Population size	$\|P\|$	-	20
Mutation probability	p_m	-	0.05
Local Search probability	p_ls	-	0.1

Among the MAs, MA_{hOX} results in the best average objective values on all but two instances. Furthermore, it finds the currently best known solution for each instance, except instance 400-09. However, due to the heuristic crossover operator which requires to maintain the information about violated slots during the search, the solution time of MA_{hOX} as well as MA_{NCPX} is considerably larger than MA_{OX}. Compared to MA_{NCPX}, MA_{OX} leads to better average objective values on 5 instances and inferior values on 3 instances. Thus, maintaining adjacency relationships between parent solutions during crossover, as in MA_{OX} and MA_{hOX}, seems promising. Given that the differences in the solution quality of all MAs are not very large, the main contribution to the results is assumed to stem from the VNS operator and not the recombination operator. This is also confirmed by the results of the VNS_1 algorithm, which shows the overall best performance. It requires considerably less time than the MA algorithms and finds the best known solutions for all instances and even improved solutions with 3 and 27 violations for instances 200-03 and 300-05, respectively. Considering the average objective values, VNS_1 leads to the best results on 38 out of 40 instances. When changing the applied neighborhood operators and their probabilities, as in VNS_2, the solution quality on the instances of set two decreases.

VI. CONCLUSIONS

We analyze the fitness landscapes of a set of CS instances by measuring the autocorrelation as well as fitness-distance correlation when four different neighborhood operators and three distances are applied. The results show a smooth landscape in terms of the normalized correlation length for the reverse, swap and shift neighborhood. The adjacency distance is suitable to describe structural relations between solutions as it leads to a high fitness-distance correlation. Furthermore, a big valley structure can be identified when using the adjacency distance metric. The findings are included in two metaheuristics for CS, a variable neighborhood search (VNS) and a memetic algorithm (MA) evaluated with three different crossover operators. In experiments, we show the superiority of the VNS algorithm as it finds and even improves currently best known solutions for instances of the CSPLib. Despite that the performances of the MA algorithms are inferior compared to VNS, MAs with a crossover operator that respects the adjacency distance metric have a better solution quality than MAs without.

The findings of the fitness-distance correlation analysis would be more meaningful if the true distances according to the applied neighborhood operators could be used. Therefore, future research should address efficient algorithms or at least good approximations to determine these distances. Further insights should also be gained as to why local search in general leads to better results than EAs for the CS problem. We assume redundant solutions to have a negative effect on the solution quality of EAs. Other representations for CS or the normalization of CS solutions prior to recombination should be analyzed and incorporated in EAs for CS.

REFERENCES

[1] B. Parrello, W. Kabat, and L. Wos, "Job-Shop Scheduling Using Automated Reasoning: A Case Study of the Car-Sequencing Problem," *Journal of Automated Reasoning*, vol. 2, pp. 1–42, 1986.

[2] T. Kis, "On the complexity of the car sequencing problem," *Operations Research Letters*, vol. 32, pp. 331–335, 2004.

[3] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*, 1995, pp. 184–192.

[4] L. Wester and M. D. Kilbridge, "The Assembly Line Model-Mix Sequencing Problem," in *Proceedings of the Third International Conference on Operations Research*, 1964.

[5] M. Gravel, C. Gagne, and W. L. Price, "Review and comparison of three methods for the solution of the car sequencing problem," *Journal of the Operational Research Society*, vol. 56, pp. 1287–1295, 2005.

[6] N. Boysen, M. Fliedner, and A. Scholl, "Sequencing mixed-model assembly lines: Survey, classification and model critique," *European Journal of Operational Research*, vol. 192, no. 2, pp. 349–373, 2009.

[7] C. Solnon, V. Cung, A. Nguyen, and C. Artigues, "The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 912–927, 2008.

[8] M. Prandtstetter and G. Raidl, "An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1004–1022, 2008.

[9] M. Fliedner and N. Boysen, "Solving the car sequencing problem via Branch & Bound," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1023–1042, 2008.

[10] U. Golle, F. Rothlauf, and N. Boysen, "Iterative Beam Search for Car Sequencing," Department of Information Systems, University Mainz, Tech. Rep., 2011.

[11] J. Gottlieb, M. Puchta, and C. Solnon, "A study of Greedy, Local Search, and Ant Colony Optimization Approaches for Car Sequencing Problems," in *Applications of Evolutionary Computing*, ser. LNCS, S. Cagnoni, C. Johnson, J. Cardalda, E. Marchiori, D. Corne, J.-A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G. Raidl, and E. Hart, Eds., vol. 2611. Springer Berlin / Heidelberg, 2003, pp. 246–257.

[12] M. Butaru and Z. Habbas, "Solving the Car-Sequencing Problem as a Non-binary CSP," in *Principles and Practice of Constraint Programming - CP 2005*, ser. LNCS, vol. 3709. Springer Berlin / Heidelberg, 2005, p. 840.

[13] M. Puchta and J. Gottlieb, "Solving Car Sequencing Problems by Local Optimization," in *Applications of Evolutionary Computing*, ser. LNCS, S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, and G. Raidl, Eds., vol. 2279. Springer Berlin / Heidelberg, 2002, pp. 132–142.

[14] L. Perron and P. Shaw, "Combining Forces to Solve the Car Sequencing Problem," in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, ser. LNCS 3011, J.-C. Regin and M. Rueher, Eds. Springer, 2004, pp. 225–239.

[15] B. Estellon, F. Gardi, and K. Nouioua, "Two local search approaches for solving real-life car sequencing problems," *European Journal of Operational Research*, vol. 191, no. 3, pp. 928–944, 2008.

[16] J.-F. Cordeau, G. Laporte, and F. Pasin, "Iterated tabu search for the car sequencing problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 945–956, 2008.

[17] T. Warwick and E. Tsang, "Tackling Car Sequencing Problems Using a Generic Genetic Algorithm," *Evolutionary Computation*, vol. 3, no. 3, pp. 267–298, 1995.

[18] A. Jaszkiewicz, P. Kominek, and M. Kubiak, "Adaptation of the genetic local search algorithm to a car sequencing problem," in *7th National Conference on Evolutionary Algorithms and Global Optimization*, Kazimierz Dolny, Poland, 2004, pp. 67–74.

[19] J. Terada, H. Vo, and D. Joslin, "Combining Genetic Algorithms with Squeaky-Wheel Optimization," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, ser. GECCO '06. New York, NY, USA: ACM, 2006, pp. 1329–1336.

[20] A. Zinflou, C. Gagne, and M. Gravel, "Crossover Operators for the Car Sequencing Problem," in *Proceedings of the 7th European Conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP 2007, Valencia, Spain, April 11-13, 2007.*, 2007, pp. 229–239.

[21] C. Gagne, M. Gravel, and W. Price, "Solving real car sequencing problems with ant colony optimization," *European Journal of Operational Research*, vol. 174, pp. 1427–1448, 2006.

[22] C. Solnon, "Combining two pheromone structures for solving the car sequencing problem with Ant Colony Optimization," *European Journal of Operational Research*, vol. 191, no. 3, pp. 1043–1055, 2008.

[23] A. Zinflou, C. Gagne, and M. Gravel, "Designing hybrid integrative evolutionary approaches to the car sequencing problem," in *IPDPS 2008. IEEE International Symposium on Parallel and Distributed Processing*, 2008, pp. 1–8.

[24] P. Merz and B. Freisleben, "Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem," *IEEE Transactions On Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, 2000.

[25] E. D. Weinberger, "Correlated and uncorrelated fitness landscapes and how to tell the difference," *Biological Cybernetics*, vol. 63, no. 5, pp. 325–336, 1990.

[26] P. F. Stadler, "Landscapes and their correlation functions," *Journal of Mathematical Chemistry*, vol. 20, no. 1, pp. 1–45, 1996.

[27] H. Hoos and T. Stützle, *Stochastic Local Search*. Amsterdam: Elsevier, 2005.

[28] T. Schiavinotto and T. Stützle, "A review of metrics on permutations for search landscape analysis," *Computers & Operations Research*, vol. 34, no. 10, pp. 3143–3153, 2007.

[29] C. Reeves, "Landscapes, operators and heuristic search," *Annals of Operations Research*, vol. 86, pp. 473–490, 1999.

[30] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[31] F. Glover, "Tabu Search - Part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[32] N. Mladenovic and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097 – 1100, 1997.

[33] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, 2nd ed. Springer, 2006.

[34] S.-S. Choi and B.-R. Moon, "Normalization for genetic algorithms with nonsynonymously redundant encodings," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 5, pp. 604 – 616, 2008.

[35] F. Rothlauf, "On the Locality of Representations," in *Genetic and Evolutionary Computation — GECCO 2003*, ser. LNCS, E. Cantú-Paz, Ed., vol. 2724. Berlin Heidelberg: Springer, 2003, pp. 1608–1609.

[36] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," Caltech Concurrent Computation Program, Tech. Rep., 1989.

[37] C. Cotta and A. Fernandez, "Memetic Algorithms in Planning, Scheduling, and Timetabling," in *Evolutionary Scheduling*, ser. Studies in Computational Intelligence, K. Dahal, K. Tan, and P. Cowling, Eds. Springer Berlin / Heidelberg, 2007, vol. 49, pp. 1–30.

[38] E. Burke and J. Landa Silva, "The Design of Memetic Algorithms for Scheduling and Timetabling Problems," in *Recent Advances in Memetic Algorithms*, ser. Studies in Fuzziness and Soft Computing, W. Hart, J. Smith, and N. Krasnogor, Eds. Springer Berlin / Heidelberg, 2005, vol. 166, pp. 289–311.

[39] P. Moscato and C. Cotta, "A Gentle Introduction to Memetic Algorithms," in *Handbook of Metaheuristics*, ser. International Series in Operations Research & Management Science. Springer, 2003, vol. 57, pp. 105–144.

[40] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.

[41] N. J. Radcliffe, "Forma Analysis and Random Respectful Recombination," in *ICGA*, 1991, pp. 222–229.

[42] G. Sywerda, "Uniform crossover in genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 2–9.

[43] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proceedings of the 9th international joint conference on Artificial intelligence - Volume 1*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1985, pp. 162–164.

[44] L. D. Whitley and J. Kauth, "GENITOR: A different genetic algorithm," in *Proc. Rocky Mountain Conf. Artificial Intel.*, Denver, CO, 1988, p. 118–130.