EURO J Comput Optim (2014) 2:279–296
DOI 10.1007/s13675-014-0023-6

ORIGINAL PAPER

A Lagrangian-ACO matheuristic for car sequencing

Dhananjay Thiruvady · Andreas Ernst · Mark Wallace

Received: 2 April 2013 / Accepted: 17 June 2014 / Published online: 10 July 2014

Abstract In this study, we investigate a hybrid Lagrangian relaxation ant colony optimisation for optimisation version of the car sequencing problem. Several cars are required to be scheduled on an assembly line and each car requires a number of options such as sunroof and/or air conditioning. These cars are required to be sequenced such that sub-sequences of specific sizes may only include a limited number of any option. While this is usually a hard constraint, in this study we treat it as a soft constraint and further require the utilisation of options be modulated across the sequence leading to the optimisation problem. We investigate various Lagrangian heuristics, ant colony optimisation (ACO) and hybrids of these methods. The results show that the Lagrangian-ACO hybrid is the best performing method for up to 300 cars.

Keywords Ant colony optimisation · Lagrangian relaxation · Hybrid methods · Car sequencing

Mathematics Subject Classification 90B99

D. Thiruvady (✉) · A. Ernst
CSIRO Computational Informatics, Melbourne, Australia
e-mail: Dhananjay.Thiruvady@csiro.au

A. Ernst
e-mail: Andreas.Ernst@csiro.au

D. Thiruvady
Australia and Clayton School of Information Technology, Monash University, Melbourne, Australia

M. Wallace
Caulfield School of Information Technology, Monash University, Caulfield East, Australia
e-mail: Mark.Wallace@monash.edu

Introduction

Hybrid methods involving meta-heuristics and exact methods have recently proved to be effective for combinatorial optimisation (Blum et al. 2008; Glover and Kochenberger 2003; Maniezzo et al. 2010; Puchinger and Raidl 2005). Recently, hybrids of meta-heuristics and mathematical programming have become popular with dedicated workshops and journals. Meta-heuristics are known to be effective on problems with large search spaces but often break-down when dealing with non-trivial hard constraints. Such problems are typical of those that appear in the real world and to tackle such problems the traditional approach has been to employ penalty-based methods (Coello 2002). Such methods typically require problem-specific tuning and can suffer from performance issues. More recently, hybrids involving meta-heuristics with constraint and integer programming have demonstrated the effectiveness of such integrations (Ernst 2010; Meyer and Ernst 2004; Puchinger and Raidl 2004; Thiruvady et al. 2009, 2012). In this paper we demonstrate how an effective matheuristic can be created by combining Lagrangian relaxation and ant colony optimisation (ACO) and apply this hybrid method to the car sequencing problem.

Lagrangian relaxation (LR) is a technique that has been used effectively with integer programming problems (Fisher 2004). This technique is based on identifying constraints in the integer programming formulation, which if relaxed, results in a simpler problem which maybe solved efficiently. The solution to the relaxed problem provides bounds on the original problem, allowing performance guarantees on the quality of feasible solutions obtained. In a minimisation problem, the relaxed problem provides lower bounds. The LR method adds a penalty to the objective for all components where the relaxed constraint is violated. By appropriately modifying the penalties, the algorithm is guided towards feasible regions.

Ant colony optimisation (ACO) was proposed by Dorigo (1992) for combinatorial optimisation. This procedure is based on the foraging behaviour of real ants. When ants look for food they mark the paths they use with a chemical called pheromones. These pheromone deposits are used by other ants in the future when looking for food by favouring those paths with a higher amount of pheromone. These ants, in turn, deposit pheromones thereby creating a positive feedback loop leading to ants converging on better paths to food sources over-time (Camazine et al. 2001). Using these principles, an algorithm can be designed to solve combinatorial optimisation problems and several studies have shown how ACO can be successfully designed for a particular problem (Dorigo and Stűtzle 2004). Various hybrids involving ACO have been investigated. These include integrations with other meta-heuristics (Blum 2005; Blum et al. 2008; López-Ibáñez et al. 2009) and exact methods such as integer programming (Anghinolfi et al. 2011) and constraint programming (Khichane et al. 2008; Meyer and Ernst 2004; Thiruvady et al. 2009, 2012). An LR–ACO hybrid was examined on the multi-choice multi-dimensional problem (Ren et al. 2012). This study used the LR solution information as a guide in the ACO procedure through the heuristic information. A second study considers Lagrangian relaxation within ACO for an optimal overlay communication network in peer-to-peer computing (Maniezzo et al. 2004).

We consider the car sequencing problem of scheduling cars on an assembly line (Dincbus et al. 1988; Gent 1998; Hentenryck et al. 1992; Parrello et al. 1986). The cars require one or many options such as air conditioning, sunroof, radio, etc. Each option is installed at specialised stations which may only handle a limited number of cars at a time. The aim of this problem is to produce a sequence (or permutation) of the cars such that when they pass through the stations the station's capacities are not exceeded. Thus, car sequencing is a satisfiability problem and determining the sequence is known to be NP-hard (Kis 2004).

Approaches to car sequencing as a satisfiability problem include constraint and integer programming (Dincbus et al. 1988; Gravel et al. 2004; Hentenryck et al. 1992). Dincbas et al. (1988) considered a pure constraint programming (CP) approach and showed that it is very effective. Their algorithm was able to sequence as many as 200 cars within 5 min. This was also the case for instances where the utilisation of options was up to 90 %. Gravel et al. (2004) developed an integer programming approach based on violation of the constraints where the objective was to minimise the number of violations.

Here, we investigate the optimisation version of the car sequencing problem (Bautista et al. 2008). In previous studies with the optimisation problem, the focus was still on feasibility where once feasibility is obtained, then the modulation of utilisation of sub-sequences can be considered (Bautista et al. 2008; Thiruvady et al. 2011). Bautista et al. (2008) consider a Beam search (Blum 2005; Valente et al. 2008) approach and make use of effective lower bounds to expand the search trees. This study effectively solves problems with 100 cars but their algorithm struggles to find feasibility for instances with 200 or more cars. Thiruvady et al. (2011) use a hybrid approach based on ACO, Beam search and constraint programming (Marriott and Stuckey 1998). Compared to the pure feasibility problem, the aim in this study is to trade-off feasibility for utilisation of options within sub-sequences. The algorithm suggested in this study is currently the best performing method for the optimisation version of car sequencing.

In this study, we aim to determine good solutions (upper bounds) to a number of hard problem instances known in the literature. We investigate LR and ACO approaches including hybrids of these methods.[1] We propose two integrations, one being a relatively loose integration where ACO is used to improve the upper bounds. Secondly, by keeping a history of the solutions obtained by the LR method, the pheromone trails of ACO are initialised with this history which are used to improve the best LR solution with ACO. In principle, such integrations can be devised with any integer program where relaxed problem solutions may be used to guide ACO. For example, solution information obtained from a root relaxation can be used to guide ACO towards feasible and/or optimal regions.

This paper is organised as follows. The problem is formally stated in Sect. 2 including the original version and the integer programming formulation of optimisation version to be tackled here. Section 3 discusses potential relaxations and provides the outline of the LR heuristic. This section also describes ACO and how ACO and LR can be combined. Section 5 details the experiments carried out and the results that follow with a discussion of the results. Section 6 concludes this paper.

[1] Note that the hybrids proposed here are different to the one suggested in Ren et al. (2012).

Problem definition

The formal definition of the car sequencing problem is as follows, based on Solnon et al. (2008). A number of cars D and a number of options O are given. Each car requires a number of these options specified by $r_{ij} \in \{0, 1\}, \forall i \in \{1, \ldots, D\}, j \in \{1, \ldots, O\}$ where $r_{ij} = 1$ states that car i requires option j. We are also given $(p_j, q_j), j \in \{1, \ldots, O\}$ which impose constraints that require that at most p_j cars may use option i in a subsequence of length q_j satisfying the capacity of a station. The cars are grouped into C classes with d_i cars per class $i \in \{1, \ldots, C\}$. Every car in the same class requires the same options such that $k, l \in d_i, j \in \{1, \ldots, O\} : r_{kj} = r_{lj}$. Hence with a slight abuse of notation we will write r_{ij} when i is a class of cars. A solution to the problem may be represented by a permutation π. There are n decision variables $\pi_1, \ldots, \pi_n$ where each variable $\pi_i \in \{1, \ldots, C\}$ such that the option constraints are satisfied.

Like any satisfiability problem, the car sequencing problem can be framed as an optimisation problem. Essentially, for constructive methods, the objective is to maximise the number of decision variables assigned such that constraints are satisfied. The search terminates when all decision variables are assigned, i.e., all cars are sequenced. Additionally, departing from satisfiability only, Bautista et al. (2008) considers further aspects of the problem which can be optimised. They define two different but related measures based on feasibility and how well the options are modulated across sub-sequences.

The first measure is the *upper over-assignment* of a sequence which is the number of times an option appears over the allowed number when the sub-sequence constraint is violated:

$$\mathrm{uoa}(\pi) = \sum_{i=1}^{O} \sum_{j=p_i}^{D} a_{ij} y_{ij}(\pi) \tag{1}$$

where for $j \geq p_i$

$$y_{ij}(\pi) = \max \left\{ 0, -p_i + \sum_{k=u_i(j)}^{j} r_{\pi_k i} \right\} \tag{2}$$

and $u_i(j) = \max(1, j+1-q_i)$. The a_{ij} terms can be seen as penalties for violating the constraints. Essentially, the sum in the equation above adds up the number of options that are used in a subsequence from $j+1-q_i$ to j. Of course, if we have not sequenced q_i cars yet we only consider the smaller size subsequence. $\mathrm{uoa}(\pi) = 0$ implies that a feasible solution to the problem has been found since we are only summing over the quantity of the violations. Similarly, *upper under-assignment* of a sequence can be defined as

$$\mathrm{uua}(\pi) = \sum_{i=1}^{O} \sum_{j=p_i}^{D} b_{ij} z_{ij}(\pi) \tag{3}$$

where

$$z_{ij}(\pi) = \max \left\{ 0, p_i - \sum_{k=u_i(j)}^{j} r_{\pi_k i} \right\} \tag{4}$$

and $u_i(j)$ is as defined earlier. Via b_{ij} we are able to modulate the usage of options in the sub-sequences. For our experiments, we randomly select a_{ij} and $b_{ij} \in (0, 1]$. This in effect imposes preferences on those sub-sequences where all the allowed options should be sequenced or as many of them as possible. Given these two measures, we can define the optimisation version of the problem to minimise

$$\text{uoa}(\pi) + \text{uua}(\pi) \tag{5}$$

Integer programming formulation

Bautista et al. (2008) provided an integer programming formulation which we make use of in this study. In addition to the y and z variables, we use x_{it} to represent a car class i at position t. For further details please refer to Bautista et al. (2008).

$$\min \sum_{j=1}^{O} \sum_{t=p_j}^{D} (a_{jt} y_{jt} + b_{jt} z_{jt}) \tag{6}$$

subject to

$$\sum_{i=1}^{C} x_{it} = 1 \qquad \forall t = 1, \ldots, D \tag{7}$$

$$\sum_{t=1}^{D} x_{it} = d_i \qquad \forall i = 1, \ldots, C \tag{8}$$

$$\sum_{i=1}^{C} \sum_{k=l_j(t)}^{t} \bar{r}_{ij} x_{ik} = p_j + y_{jt} - z_{jt}$$

$$\forall j = 1, \ldots, O;\ \forall t = p_j, \ldots, D; \tag{9}$$

$$z_{jt}, y_{jt} \geq 0 \forall j = 1, \ldots, O;\ \forall t = p_j, \ldots, D \tag{10}$$

$$x_{it} \in \{0, 1\} \forall i = 1, \ldots, C;\ \forall t = 1, \ldots, D \tag{11}$$

where $l_j(t) = \max\{1, t + 1 - q_j\}$.

Lagrangian relaxation

There are three potential relaxations for the above formulations, i.e., relaxing Eqs. (7), (8) or (9). Relaxing Eq. (7) yields the following objective subject to the other remaining constraints:

$$\text{LLR}_1(\lambda) = \min \sum_{j=1}^{O} \sum_{t=1}^{D} (a_{jt} y_{jt} + b_{jt} z_{jt}) + \sum_{t=1}^{D} \lambda_t \sum_{i=1}^{C} (x_{it} - 1) \tag{12}$$

Here, we have a Lagrangian multiplier or penalty associated with every position and this relaxation allows more than one car to be sequenced at a position. Now relaxing

Eq. (8) yields:

$$\mathrm{LLR}_2(\lambda) = \min \sum_{j=1}^{O} \sum_{t=1}^{D} (a_{jt} y_{jt} + b_{jt} z_{jt}) + \sum_{i=1}^{C} \lambda_i \sum_{t=1}^{D} (x_{it} - d_i) \tag{13}$$

where a penalty for each car class exists. In this formulation we may have more or less than the allowed cars per class. Finally, Eq. (9) yields:

$$\begin{aligned}\mathrm{LLR}_3(\lambda) = \min & \sum_{j=1}^{O} \sum_{t=1}^{D} (a_{jt} y_{jt} + b_{jt} z_{jt}) \\ & + \sum_{j=1}^{O} \sum_{t=1}^{D} \lambda_{jt} \left(z_{jt} - y_{jt} + \sum_{i=1}^{C} \sum_{k=l_j(t)}^{t} r_{ij} x_{ik} - p_j \right)\end{aligned} \tag{14}$$

which can be re-arranged to obtain:

$$\begin{aligned}\mathrm{LLR}_3(\lambda) = \min & \sum_{j=1}^{O} \sum_{t=p_j}^{D} \left[y_{jt}(a_{jt} - \lambda_{jt}) + z_{jt}(b_{jt} - \lambda_{jt}) - p_j \lambda_{jt} \right] \\ & + \sum_{i=1}^{C} \sum_{k=1}^{D} \left(\sum_{j=1}^{O} r_{ij} \sum_{t=p_j}^{\min\{D, k+q_j\}} \lambda_{jt} \right) x_{ik}\end{aligned} \tag{15}$$

To ensure that the Lagrangian dual remains finite, the multipliers have to be constrained to be in some interval, for example, $-b_{jt} \leq \lambda_{jt} \leq a_{jt}$.

We carried out a preliminary investigation into each of these alternative Lagrangian relaxations and found the first to be most effective. The problem in Eq. (13) requires large time overheads compared to the problems in Eqs. (12) or (15). Additionally, the problem in Eq. (15) was solved very quickly but did not provide effective lower bounds. Hence, for this study we focus on the problem associated with Eq. (12) and in the remainder of this paper we shall refer to $\mathrm{LLR}(\lambda) = \mathrm{LLR}_1(\lambda)$.

The Lagrangian heuristic

A Lagrangian heuristic (Boschetti and Maniezzo 2009) for car sequencing can be defined as follows. An LR algorithm is presented in Algorithm 1. Here we use the model obtained from relaxing Eq. (12). The algorithm is first initialised with various parameters and multipliers. The main loop starts at line 5 and executes for 1,000 iterations while the gap and γ are above a specified threshold. Each procedure is described in detail below.

The relaxed problem is solved in Solve(λ^i, LB). This involves minimising the Lagrangian function, i.e., Eq. (12) subject to the remaining constraints (8)–(11). The lower bound LB is set to

$$\mathrm{LB} = \mathrm{LB}(\mathrm{LLR}(\lambda^i)) \tag{16}$$

Algorithm 1 LR for the car sequencing problem

```
1: INPUT: A car sequencing instance
2: $\pi^{bs}$ := NULL (best solution)
3: initialize $\lambda_t^0 = 0, \forall t \in \{1, \dots, D\}$
4: $\gamma := 2.0$, $k := 0$, $gap := \infty$, $UB^* := \infty$, $LB^* := -\infty$
5: while $\gamma > 0.01$ & $gap > 0.01$ & $i < 1000$ do
6:     $x$ = Solve($\lambda^i$, $LB$)
7:     $\pi$ = GenerateSequence($x$)
8:     ImproveUB($\pi$)
9:     UpdateBest($\pi^{bs}$,$\pi$,$\gamma$)
10:    $LB* = f(\pi^{bs})$
11:    UpdateMult($\lambda^i$, $LB^*$, $UB^*$, $x$, $\gamma$)
12:    $gap = \frac{UB^*-LB^*}{UB^*}$
13:    i ← i + 1
14: end while
15: OUTPUT: $\pi^{bs}$
```

which is a lower bound on LLR(λ).[2] The above procedure returns a sequence of cars, where more than one may be sequenced at a single position and none sequenced at others.

A sequence of cars can be obtained from x in a straight-forward manner (GenerateSequence(x)). For each position t, consider whether a car class has been sequenced here and if it has, then append this car class to π. If there are multiple car classes in position t, they can be sequenced in different ways including the order in which they are found or randomly. We test both these options here. By continuing this for the entire sequence, a complete sequence of the cars is obtained.

The sequence π can be improved by various methods in the procedure ImproveUB (π). For example, local search may be used to improve these solutions. The best solution obtained using an improvement or not is updated using UpdateBest (π^{bs},π,γ) which does two things. Firstly, $\pi^{bs} = \pi$ if $f(\pi) < f(\pi^{bs})$. Secondly, if π^{bs} has not been updated in the last ten iterations, $\gamma \leftarrow \gamma \div 2$.

Finally, the procedure UpdateMult(λ^i, LB*, UB*, x, γ) uses subgradient optimisation (Bertsekas et al. 2003) to update the multipliers for all positions $t \in \{1, \dots, D\}$, $k \in R$

$$\lambda_t^{i+1} = \lambda_t^i + \frac{\gamma(\mathrm{UB}^* - \mathrm{LB}^*)\Delta_t}{\sum_{\hat{t}=1}^{D} \Delta_{\hat{t}}^2} \tag{17}$$

where $\Delta_t = \sum_{i=1}^{P} x_{it} - 1$ and x are the variables from the model in Sect. 2.1.

Ant colony optimisation

We use the ACO implementation in Thiruvady et al. (2011) with minor differences which are detailed below. For the sake of completeness we provide the algorithm and the relevant details. ACS is the variant of ACO used in this study (Dorigo and Stűtzle 2004) and is presented in Algorithm 2. The pheromones τ_{ij} represent the desirability

[2] Since LLR(λ) is itself a hard problem to solve, we allow it to be solved to a gap or a time limit. To ensure a valid lower bound, we use the lower bound of LLR(λ) as the final lower bound.

Algorithm 2 ACO for the car sequencing problem

```
1: INPUT: A car sequencing instance, $\mathcal{T}$, $\pi^{bs}$
2: while termination conditions not satisfied do
3:    $S := \emptyset$
4:    for $j = 1$ to $n_{ants}$ do
5:       $\pi_j :=$ ConstructSequence()
6:       $S := S \cup \{\pi_j\}$
7:    end for
8:    $\pi^{ib} := argmin\{f(\pi)|\pi \in S\}$
9:    $\pi^{bs} =$ Update($\pi^{ib}$)
10:   $\mathcal{T}$ = PheromoneUpdate($\pi^{bs}$)
11: end while
12: OUTPUT: $\pi^{bs}$
```

of picking car class j in position i of the sequence (i.e., $\pi_i = j$). There is no distinction between cars within the same class.

A car sequence π is constructed by incrementally adding car classes to π (ConstructSequence()). A solution is complete when all cars from all classes have been sequenced. A car class is selected as follows. A random number $q \in (0, 1]$ is generated and compared to the parameter q_0. For the ith variable, π_i, if $q < q_0$, a car class k is deterministically selected at variable i according to

$$k = \underset{j \in \mathcal{C}}{\operatorname{argmax}}\, \tau_{ij} \times \eta_j \tag{18}$$

otherwise, $\pi_i = k$ is selected with probability

$$\mathbf{p}(\pi_i = k) = \frac{\tau_{ik} \times \eta_k}{\sum_{j \in \mathcal{C}} \left(\tau_{ij} \times \eta_j\right)} \tag{19}$$

where η_k is a heuristic factor that may be used to bias the selection. While Thiruvady et al. (2011) use the dynamic sum of utilisation rates (Gottlieb et al. 2003) as the heuristic, we find that the latter heuristic is not as effective for this study. This heuristic essentially leads the solution construction towards feasibility[3] which is not the primary objective in this study where the aim is to achieve the ideal utilisation of the options. Ideal utilisation is an optimisation goal, and experience with local search methods has shown that improvement towards an optimal solution is often achieved by admitting infeasible candidate solutions en route to the optimum.

All solutions are stored in the set S. The solution with minimum cost ($f(\pi) = \text{uua}(\pi) + \text{uoa}(\pi)$) from this set is the iteration best (π^{ib}). If π^{ib} is an improvement over π^{bs}, π^{bs} is set to π^{ib} ($\pi^{bs} =$ Update(π^{ib})).

The pheromone trails are updated corresponding to the solution components in π^{bs} ($\mathcal{T}$ = PheromoneUpdate(π^{bs})) as follows

$$\tau_{ij} = \tau_{ij} \times \rho + \delta \tag{20}$$

[3] Note that feasibility requires that $y_{ij} = 0, \forall i \in \mathcal{V}, j \in \mathcal{O}$.

where $\delta = \hat{\delta}/(1.0 + f(\pi^{bs}))$, $\hat{\delta}$ is determined such that $\delta \in [0.01, 0.1]$. ρ is a learning rate which is set to be relatively large (0.1) for this study and was found through tuning by hand.

Hybrid Lagrangian-ACO matheuristic

We suggest two different hybrids of LR and ACO. The first is obtained as follows. On line 8 of the LR algorithm (Algorithm 1) we have the option of improving the upper bounds. Here, we use the ACO procedure described above leading to *LR–ACO*.[4] The modified algorithm works as follows. The current best solution and the pheromone trails are given as input. This essentially biases the search towards promising regions that have been see already in the past.[5]

To apply improvements, a feasible solution needs to be constructed from the solution to the relaxed problem. In Sect. 3.1 we show how this can be achieved. Essentially, we can start at the first position and incrementally add car classes to π where they appear in the solution to the relaxed problem. If there are two or more cars sequenced at a particular position they are appended to π by index. Alternative ways to determine π can also be used, for example, if there are two or more cars sequenced at a position, randomly select from amongst these to append to π and proceed to select randomly from the reduced number of cars.

A second hybrid is one where ACO is executed after the LR algorithm completes (*LR+ACO*). We suggest two possible schemes here. Firstly, the best LR solution can be given as input to ACO which is executed as described above. Secondly, a history of the best solutions could be kept which can be used to provide ACO with a pheromone matrix which has already partially converged. This would allow ACO to bias its search to all promising areas seen in the past.

Determining the pheromone matrix in second LR+ACO scheme can be done as follows. The matrix is initialised as follows $\tau_{ij} = 1.0\,\forall i \in 1, \ldots, D$ and $j \in 1, \ldots, C$. During every iteration, $\forall i, j$: $\tau_{ij} \leftarrow \tau_{ij} + x_{ij}$. This update essentially favours picking sequences according to the solution of the relaxed problem. Note that there is no bias introduced if two car classes were selected at a single position as they are both favoured equally. When the LR scheme is complete, the matrix is normalised so that the distribution for each variable in the sequence sums to 1.0. It is well known that under appropriate conditions the average of the primal solutions to the Lagrangian subproblems converges to an optimal LP relaxation solution, see Kurt et al. (2009). Hence, this approach effectively uses an approximation of the fractional LP solution as a starting value for the pheromones.

Considering the two LR+ACO schemes suggested above, we find that the second option provides improved results. Hence, LR+ACO refers to the second implementation in the remainder of this paper.

[4] Note that this may be done in several ways including a local search algorithm.

[5] This scheme is similar to the one proposed by Ren et al. (2012). While we could incorporate the LR solution information in ACO as a heuristic bias, we found that this was not so effective.

Experiments and results

Experiments were conducted with the LR heuristic (also a randomised version), ACO, LR−ACO and LR+ACO on instances obtained from Gent and Walsh (1999), Gravel et al. (2004), and Perron and Shaw (2004). Additional experiments were conducted with a subset of instances from Perron and Shaw (2004) which is considered to be one of the datasets with the hardest instances to solve (Khichane et al. 2008). To further understand the algorithms, we consider a subset from Perron and Shaw (2004) which is the same subset of instances used by Thiruvady et al. (2011). Here, we also compare the lower bounds obtained by the LR-based methods to the ILP root relaxation of the problem obtained by Gurobi. This study generated a_{ij} and b_{ij} values in the interval (0, 1] and we generate these values in exactly the same fashion. Using the same seed, these values are generated once at the beginning of each run. Hence for each instance, the values are always equal.

For the complete set of runs we conducted one run per instance. However, for the subset of instances we conducted 30 runs per instance for all ACO-based algorithms to demonstrate significance of the results obtained. Each run was given 2 h of CPU time and the experiments were conducted on three resources. The first was a quad core i5-2400 3.10 GHz machine running Linux 64 bit operation system and the second and third resources were two clusters including the Monash Sun Grid and the Enterprisegrid using Nimrod (Abramson et al. 2000). To solve the relaxed problem, Gurobi 5.0 (Gurobi Optimization 2010) was used. Note that Eq. (12) was relaxed in all the results to follow.

The parameters for ACO and the LR hybrids with ACO were set by considering the previous study (Thiruvady et al. 2011) and through tuning by hand. The number of solutions per iteration, n_a, was set to 10. ρ was chosen as a high learning rate equal to 0.1 and $q_0 = 0.9$ was chosen to favour high deterministic selection.

Results

A single run was conducted per instances over all instances from Gent and Walsh (1999), Gravel et al. (2004) and Perron and Shaw (2004). First we examine the upper bounds by dataset (see Fig. 1). The results are averaged for all instances within the dataset and % difference to the best performing algorithm is computed. LR+ACO performs best across all the datasets and all the LR-based algorithms outperform ACO on its own. This demonstrates that ACO is greatly aided by the relaxed problem solutions provided by LR.

Secondly, we examine lower bounds by dataset (see Fig. 2). Here, LR is the best performing algorithm and while the algorithm with the ACO components is competitive, clearly the CPU cycles used up by the ACO components result in fewer iterations for the lower bound to improve.

From amongst the datasets considered above, Perron and Shaw (2004) are known to contain the hardest instances (Khichane et al. 2008). Furthermore, since ACO is stochastic, several runs with the ACO algorithms are required to determine statistical significance. Thus, we consider a subset of instances (those from Thiruvady et al.

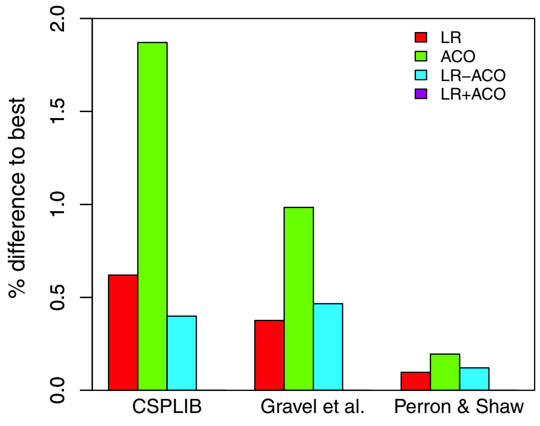

Fig. 1 A comparison of the upper bounds obtained by LR, ACO, LR−ACO and LR+ACO. The *graph* shows the % difference obtained by each algorithm across all runs on every instances compared to the best solution obtained for that instance. *Note*, LR+ACO does not appear since it provides the best results for all the instances

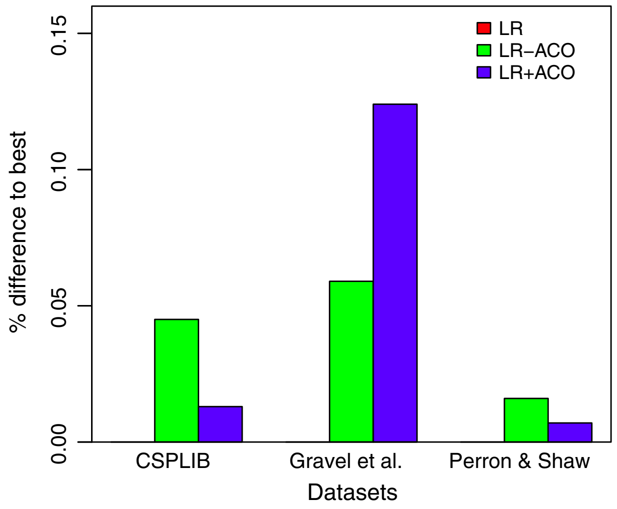

Fig. 2 A comparison of the lower bounds obtained by LR, LR−ACO and LR+ACO. There are no lower bounds associated with ACO. The *graph* shows the % difference obtained by each algorithm across all runs on every instances compared to the best solution obtained for that instance. *Note*, LR does not appear since it provides the best results for all the instances

(2011)) and repeat these experiments in the following sections. We also compare our algorithms to CP-Beam-ACO of Thiruvady et al. (2011).

We analyse upper and lower bounds for the subset of instances provided by all algorithms in the following sections. In the tables to follow, the first column shows the instance number. Lower bounds (LB), upper bound (UB) and time in terms of CPU cycles (time) are also provided where applicable. Note that for the LR-based algorithms and ACO, 2 h of run-time was allowed. The best results obtained for any instance are highlighted in boldface.

Table 1 Upper bounds obtained by the LR-based algorithms and ACO

Instance	LR	LR-rand	ACO		LR−ACO		LR+ACO	
	UB	UB	Best	Mean	Best	Mean	Best	Mean
100-22	283.5	279.28	**260.43**	270.57	260.89	265.79	267.01	281.85
100-35	250.2	250.52	225.02	233.97	**223.41**	230.8	232.75	239.9
100-64	284.71	270.1	252.78	257.83	**248.73**	256.01	262.45	262.4
100-77	208.22	200.59	174.15	180.28	**174.00**	179.29	186.04	190.32
100-82	272.48	260.56	240.06	246.94	**239.81**	249.26	244.29	251.47
100-94	206.37	196.14	182.53	189.4	**177.61**	187.14	189.24	191.49
300-8	719.11	677.59	782.82	903.75	690.06	724.25	**631.89**	653.23
300-14	898.92	886.88	990.17	996.87	882.32	933.53	**817.59**	830.58
300-53	756.77	751.61	985.83	995.2	760.99	792.79	**712.07**	716.45
300-56	651.61	632.1	705.77	810.2	650.31	684.22	**573.98**	597.84
300-62	918.32	886.3	992.73	997.25	910.28	934.72	**819.32**	839.02
300-78	742.96	700.97	780.08	951.13	706.88	752.3	**629.69**	633.03
500-14	841.4	870.45	1,041.29	1,070.61	912.29	950.69	**775.85**	776.59
500-27	1,777.1	1,719.01	1,898.05	1,924.96	1,840.19	1,857.6	**1,623.70**	1,673.9
500-65	1,084.62	1,076.87	1,202.29	1,222.22	1,078.18	1,098.82	**998.05**	998.64
500-74	991.07	958.78	1,122.82	1,146.07	1,003.83	1,042.96	**906.95**	912.57
500-79	1,748.65	1,689.54	1,780.57	1,807.41	1,650.52	1,761.08	**1,581.90**	1,582.66
500-88	1,460.42	1,374.24	1,492.12	1,516.33	1,424.76	1,482.79	**1,292.23**	1,320.94

Investigating upper bounds

Table 1 compares the upper bounds obtained by all the methods. LR-rand is similar to LR but randomises the selection of car classes at a position when there are two or more to choose from. It can be seen here that for the small problems with 100 cars, LR−ACO is the best performing algorithm. ACO finds the best solution on the instance 100-22 on one run but is on average worse than LR−ACO even for this instance. Overall, LR+ACO is the best performing algorithm by achieving the best results for every instance with 300 cars or more.

These results are also plotted in Fig. 3. Here, it can be clearly seen that while ACO is effective on the small problems it increasingly worsens in performance with increasing problem size. For 300 cars or more, LR and LR-rand improve over ACO. However, overall the hybrids LR−ACO and LR+ACO are clearly the best performing.

We compare LR+ACO to CP-Beam-ACO of Thiruvady et al. (2011). A straightforward comparison is not meaningful since this study required that the uoa values (see Sect. 2) were 0 and hence infeasible solutions were not considered at all. Thus, this study solved a slightly different problem. Additionally, the CP-based algorithms required very large run-times (15 h) and to compare LR+ACO with CP-Beam-ACO, we run LR+ACO for an increased run-time.

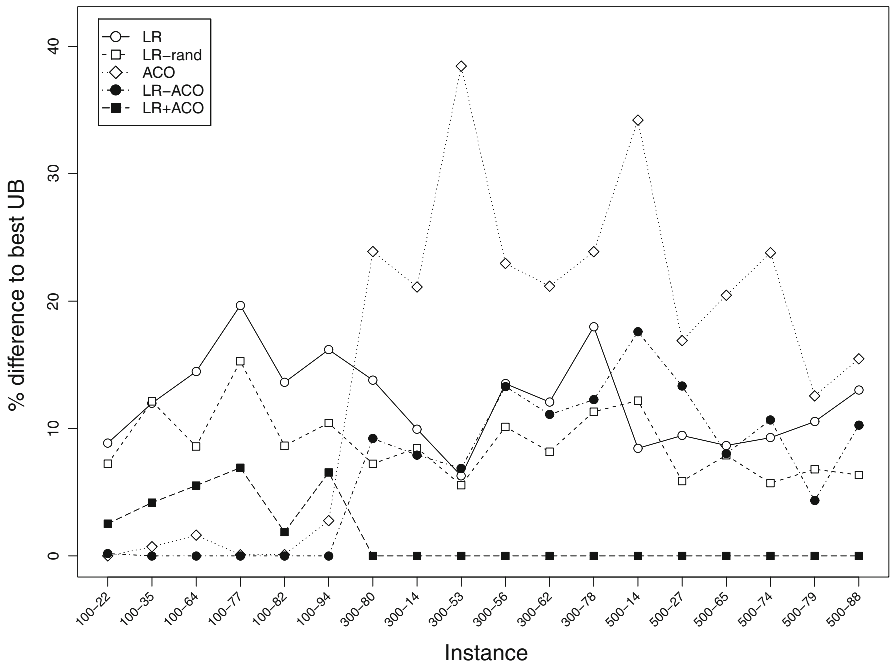

Fig. 3 A comparison of the upper bounds obtained by all methods tested in this study. The *graph* shows the % difference obtained by each algorithm across all runs on every instances compared to the best solution obtained for that instance

Table 2 presents the results for experiments with LR+ACO with additional run-times (3 h each) for the LR component and the ACO component with a total of 6 h. This table also shows the results for CP-Beam-ACO which was run for 15 h per instance. We can see here LR+ACO is the best performing algorithm for all the instances with 100 cars and nearly all the instances with 300 cars except for 300-56. Therefore, with a shorter run-time, LR+ACO is superior to CP-Beam-ACO for up to 500 cars. For the instances with 500 cars, CP-Beam-ACO fails often but is almost always superior if solutions are found.

We also conducted experiments for LR+ACO with 3 and 10 h for the LR component and ACO to compare more closely with run-times of CP-Beam-ACO. However, we still find no significant improvements for the instances with 500 cars. Thus, 300 cars appears to be the limit where ACO can improve upon the solutions obtained by the LR scheme. This is also verified by looking more closely at CP-Beam-ACO (Thiruvady et al. 2012). There are very few iterations conducted for all instances with more than 100 cars in 15 h. Thus, there is very little learning here for the ACO component and the solution quality obtained can be attributed to the CP component.

Investigating lower bounds

The lower bounds obtained are compared in Table 3. Here, LP is the bound obtained from the linear programming relaxation, Root reports root relaxation or the bounds

Table 2 Upper bound comparisons for larger run-times

Instance	LR+ACO (3 h + 3 h)		CP-Beam-ACO	
	Best	Mean	UB	Fail
100-22	**258.05**	263.1	268.95	0
100-35	**220.92**	229.69	231.16	0
100-64	**244.90**	250.55	258.35	0
100-77	**174.59**	179.58	181.37	0
100-82	**234.22**	239.29		30
100-94	**172.38**	178.58	183.56	0
300-8	**608.21**	616.72	614.42	28
300-14	**801.60**	808.73	818.12	0
300-53	**681.93**	698.27	686.07	0
300-56	553.04	562.72	534.74	0
300-62	**798.06**	805.49	838.84	2
300-78	**597.58**	617.08	610.7	16
500-14	774.78	785.69	**687.00**	0
500-27	**1,600.67**	1,618.96		30
500-65	996.95	998.5	**783.42**	29
500-74	849.01	863.29	**776.93**	2
500-79	1,505.64	1,539.91	**1,474.83**	0
500-88	**1,229.11**	1,285.95		30

determined when a single node in the search tree has been instantiated and ILP is the bound obtained when the equivalent integer program is solved to a pre-defined time limit, i.e., 720 s in these experiments. All time limits in Gurobi are based on wall clock times and hence we also report the equivalent time requirements in terms of CPU cycles. Additionally, the bounds obtained by the LR-based algorithms are also provided. It can be seen here that ILP relaxations provide the best lower bounds. The LR-based algorithms are typically superior to the LP relaxation for the smaller problems but are consistently worse for the problems with 500 cars. This indicates that close to optimal Lagrangian multipliers may be found for the smaller problems but not for the larger instances.[6]

The lower bounds are also compared in Fig. 4. For each instance, the % difference to the best lower bound obtained across all instances is plotted. The Root or ILP relaxations always provide the best lower bounds which are due to cuts generated by the Gurobi solver. Since we do not have access to the underlying methods, we are unable to investigate this more thoroughly. Experimental results using commercial integer-linear solvers have shown that even for small problems with 100 cars, the upper bounds found within a reasonable time period are uncompeti-

[6] Optimal integer solutions to the Lagrangian relaxation often provide better lower bounds than optimal solutions to the linear relaxation. Thus, if optimal multipliers are found the lower bounds of the Lagrangian will be at least as good as LP relaxation bounds.

Table 3 Lower bounds obtained by LP, Root, ILP relaxation (after 720 s of wall clock time) and the LR-based algorithms

Instance	LP		Root		ILP (720 s)	LR	LR-rand	LR−ACO	LR+ACO
	LB	Time	LB	Time	LB	LB	LB	LB	LB
100-22	233.28	1	237.27	7	**240.78**	234.60	234.4	234.11	234.26
100-35	198.30	1	200.96	18	**203.08**	199.33	198.59	198.77	199.07
100-64	222.62	1	225.67	28	**228.18**	222.57	223.3	222.56	223.11
100-77	150.50	1	153.11	5	**155.88**	151.62	150.86	151.38	151.62
100-82	206.88	1	208.90	8	**211.11**	205.66	207.24	205.73	205.66
100-94	152.94	1	155.90	6	**159.23**	153.70	153.69	153.29	153.49
300-8	482.41	1	492.26	39	**492.57**	468.93	471.73	469.51	468.93
300-14	674.30	1	682.26	40	**688.41**	675.10	675.93	667.56	675.76
300-53	580.95	1	587.43	229	**591.80**	582.12	582.17	575.13	581.31
300-56	417.69	1	428.99	424	**429.34**	419.29	416.61	410.37	418.95
300-62	679.09	1	688.31	54	**691.64**	673.66	675.9	661.47	679.09
300-78	470.45	1	478.97	720	**479.18**	459.87	463.29	457.74	468.94
500-14	489.30	2	503.32	1,125	**503.49**	491.35	488.79	471.28	491.35
500-27	1,266.03	3	1,276.66	160	**1,276.77**	1,249.09	1,251.43	1,236.99	1,244.65
500-65	588.96	2	607.19	243	**607.55**	569.66	569.66	569.66	569.66
500-74	568.05	2	585.40	161	**585.79**	558.55	561.2	551.97	558.55
500-79	1,218.18	3	1,229.03	98	**1,229.31**	1,199.98	1,201.33	1,186.37	1,199.98
500-88	935.74	2	947.60	114	**947.93**	904.26	905.14	904.32	904.26

tive. Thus, the partial solutions provided by the LR scheme are very useful for this purpose.

Conclusion

In this study we examine how to create a matheuristic by combining Lagrangian relaxation and ACO. The method is evaluated on the optimisation version of the car sequencing problem. Such hybrids have become popular recently and we show here that ACO can benefit from the accumulated information of the solution to the relaxed problems obtained from the Lagrangian heuristic. The matheuristic developed in this paper is problem independent and could be applied to any problem where ACO and/or Lagrangian relaxation methods are useful. The results across three different benchmarks show that the hybrid of LR+ACO is the best performing algorithm. Considering the more detailed analysis, we see that for small problems (100 cars), LR–ACO and ACO are competitive. However, with increasing problem size, the two-phase LR+ACO method where ACO uses the Lagrangian solutions to seed its pheromone matrix is the best performing method considering solution quality, i.e., upper bounds. Importantly the combined matheuristic performs better than either heuristic on its own.

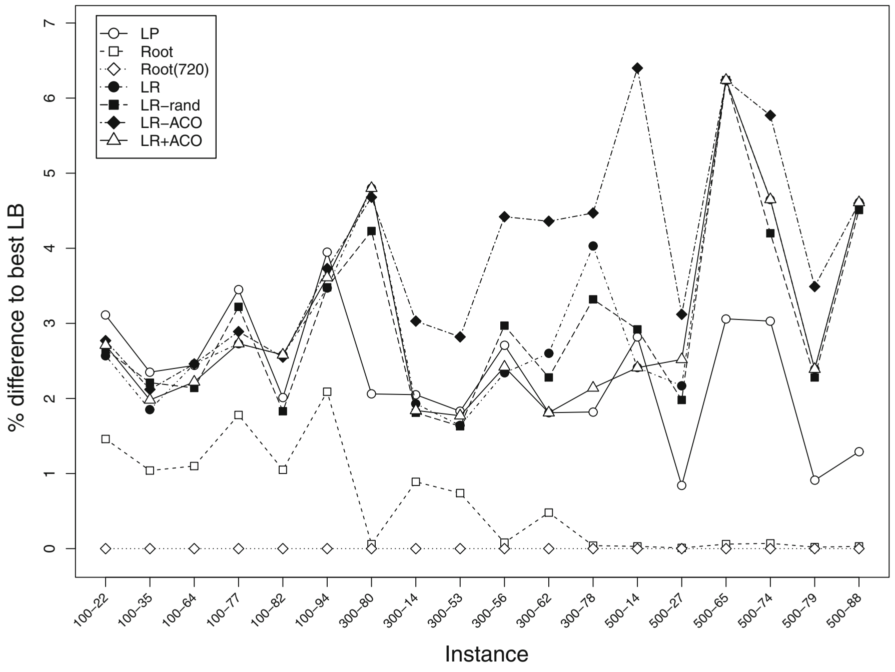

Fig. 4 A comparison of the lower bounds obtained by all algorithms relative to the best performing algorithm

Ren et al. (2012) previously investigated a hybrid of LR and ACO. They made use of the LR solution information as heuristic information in ACO. We investigated this option in this study but found that this type of hybrid is not effective here. Nonetheless, hybrids such as this one and the ones suggested in this study could be relatively easily implemented for future problems in a generic framework.

We see that the lower bounds obtained from the LR heuristics improve upon the LP relaxation for small problems but are not as competitive as the ILP relaxations. There is certainly room for improvement here. We are currently investigating methods based on cutting planes (bundle methods) and the volume algorithm to determine if these lower bounds will improve with improving search directions.

Furthermore, we have used the LR scheme to improve the upper bounds. However, we have not attempted to provide the LR scheme with information obtained from ACO. We are currently working on an implementation where the pheromone trails may be used to update the Lagrangian multipliers along the lines of the hybrid LR-PSO approach by Ernst (2010). In addition to improving the lower bounds, this could in turn help provide improved upper bounds.

Finally, we have focused on one particular relaxation in this study where more than one car class can be scheduled at a position. However, the other relaxations may also prove to be useful with further investigation. In particular, the third relaxation (see Eq. 14, Sect. 3) is solved efficiently as it amounts to solving an assignment problem. Thus, more time could be spent on the upper bounds.

References

Abramson D, Giddy J, Kotler L (2000) High performance parametric modeling with Nimrod/G: killer application for the global grid? In: International Parallel and Distributed Processing Symposium (IPDPS). IEEE Computer Society, Washington, DC, USA, pp 520–528

Anghinolfi D, Paolucci M, Sacone S, Siri S (2011) Integer programming and ant colony optimization for planning intermodal freight transportation operations. In: 2011 IEEE Conference on Automation Science and Engineering (CASE), pages 214–219

Anstreicher KM, Wolsey LA (2009) Two "well-known" properties of subgradient optimization. Math Progr 120(1):213–220

Bautista J, Pereira J, Adenso-Díaz B (2008) A beam search approach for the optimization version of the car sequencing problem. Ann Oper Res 159:233–244

Bertsekas DP, Nedić A, Ozdaglar AE (2003) Convex analysis and optimization. Athena Scientific, Cambridge

Blum C (2005) Beam-ACO: hybridizing ant colony optimization with beam search: an application to open shop scheduling. Comput Oper Res 32:1565–1591

Blum C, Blesa M, Roli A, Sampels M (eds) (2008) Hybrid metaheuristics: an emerging approach to optimization. Studies in Computational Intelligence, vol 114. Springer, Berlin

Boschetti M, Maniezzo V (2009) Benders decomposition, Lagrangian relaxation and metaheuristic design. J Heuristics 15(3):283–312

Camazine S, Deneubourg J-L, Franks NR, Sneyd J, Theraulaz G, Bonabeau E (2001) Self-organization in biological systems. Princeton University Press, Princeton

Coello C (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods in Appl Mech Eng 191:1245–1287

Dincbus M, Simonis H, Hentenryck P (1988) Solving the car-sequencing problem in constraint logic programming. In: 8th European Conference on Artificial Intelligence-ECAI 88. Pitmann Publishing, London, pp 290–295

Dorigo M (1992) Optimization, learning and natural algorithms. PhD thesis, Dip. Elettronica

Dorigo M, Stűtzle T (2004) Ant colony optimization. MIT Press, Cambridge

Ernst AT (2010) A hybrid Lagrangian particle swarm optimization algorithm for the degree-constrained minimum spanning tree problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Barcelona. IEEE, pp 1–8

Fisher M (2004) The Lagrangian relaxation method for solving integer programming problems. Manag Sci 50(12):1861–1871

Gent I (1998) Two results on car sequencing problems. Technical Report APES02. University of St. Andrews, St. Andrews, UK

Gent I, Walsh T (1999) CSPLIB: a benchmark library for constraints. Technical Report APES-09-1999. University of St. Andrews, St. Andrews, UK

Glover FW, Kochenberger GA (eds) Handbook of metaheuristics, International series in operations research and management science, vol 57. Springer, Berlin

Gottlieb J, Puchta M, Solnon C (2003) A study of greedy, local search and ACO for car sequencing problems. Lect Notes Comput Sci 2611:245–282

Gravel M, Gagné C, Price W (2004) Review and comparison of three methods for the solution of the car-sequencing problem. J Oper Res Soc 56:1287–1295

Hentenryck P, Simonis H, Dincbus M (1992) Constraint satisfaction using constraint logic programming. Artif Intell 58:113–159

Khichane M, Albert P, Solnon C (2008) CP with ACO. Lect Notes Comput Sci 5015:328–332

Kis T (2004) On the complexity of the car sequencing problem. Oper Res Lett 32:331–335

López-Ibáñez M, Blum C, Thiruvady D, Ernst AT, Meyer B (2009) Beam-ACO based on stochastic sampling for makespan optimization concerning the TSP with time windows. Lect Notes Comput Sci 5482:97–108

Maniezzo V, Boschetti M, Jelasity M (2004) An ant approach to membership overlay design. In: Dorigo M, Birattari M, Blum C, Gambardella LM, Mondada F, Stntzle T (eds) Ant colony optimization and swarm intelligence, vol 3172., Lecture Notes in Computer ScienceSpringer, Berlin, pp 37–48

Maniezzo V, Stützle T, Voß S (eds) Matheuristics—hybridizing metaheuristics and mathematical programming, Annals of information systems, vol 10. Springer, Berlin

Marriott K, Stuckey P (1998) Programming with constraints. MIT Press, Cambridge

Meyer B, Ernst A (2004) Integrating ACO and constraint propagation. Lecture notes in computer science: ant colony, optimization and swarm intelligence, vol 3172, pp 166–177

Gurobi Optimization (2010) Gurobi optimizer version 5.0. Available from: http://www.gurobi.com/

Parrello B, Kebat W, Wos L (1986) Job-shop scheduling using automated reasoning: a case study of the car sequencing problem. J Autom Reason 2:1–42

Perron L, Shaw P (2004) Combining forces to solve the car sequencing problem. In: CPAIOR-04, vol 3011. Springer, Berlin, pp 225–239

Puchinger J, Raidl GR (2004) An evolutionary algorithm for column generation in integer programming: an effective approach for 2D bin packing. Lect Notes Comput Sci 3242:642–651

Puchinger J, Raidl GR (2005) Combining metaheuristics and exact algorithms in combinatorial optimization: a survey and classification. Lect Notes Comput Sci 3562:41–53

Ren Z-G, Feng Z-R, Zhang A-M (2012) Fusing ant colony optimization with Lagrangian relaxation for the multiple-choice multidimensional knapsack problem. Inf. Sci. 182(1):15–29

Solnon C, Dat V, Cung A, Nguyen, Artigues C (2008) The car sequencing problem: overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. Eur J Oper Res 191:912–927

Thiruvady D (2012) Hybrids of stochastic metaheuristics and constraint programming for combinatorial optimization. PhD thesis, Calyton School of Information Technology

Thiruvady D, Blum C, Meyer B, Ernst AT (2009) Hybridizing beam-ACO with constraint programming for single machine job scheduling. Lect Notes Comput Sci 5818:30–44

Thiruvady D, Singh G, Ernst AT, Meyer B (2012) Constraint-based ACO for a shared resource constrained scheduling problem. Int J Prod Econ 141(1):230–242

Thiruvady DR, Meyer B, Ernst AT (2011) Car sequencing with constraint-based ACO. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO '11. ACM, New York, pp 163–170

Valente JMS, Alves RAFS (2008) Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups. Comput Oper Res 35(7):2388–2405