A Case Study of Mutual Routing-Scheduling Reformulation *

J. Christopher Beck[1], Patrick Prosser[2], and Evgeny Selensky[2]

[1] Cork Constraint Computation Center, Department of Computer Science, University College Cork, Ireland. c.beck@4c.ucc.ie
[2] Department of Computing Science, University of Glasgow, Scotland. pat/evgeny@dcs.gla.ac.uk

Abstract. Current optimisation technology performs reasonably well on classical models of scheduling and vehicle routing problems. As "impurities" are introduced, creating problems with characteristics of both scheduling and routing, algorithm choice becomes unclear: is it better to apply scheduling or vehicle routing techniques? In this paper, we undertake a systematic study of problem characteristics that differ between classical models of vehicle routing and scheduling in order to identify those that are important for the performance of typical vehicle routing and scheduling techniques. In particular, we find that the addition of temporal constraints among visits or the addition of tight vehicle specialisation constraints significantly improves the performance of scheduling techniques relative to vehicle routing techniques.

1 Introduction

Classical models of combinatorial problems such as scheduling play a key role in optimisation research. The existence of a precise and succinct problem model allows theoretical and empirical work to focus on core issues of problem solving performance. By their very nature, however, classical models are a simplification of real-world problems. As optimisation technology matures the challenges of classical models are increasingly being addressed, meaning that the aspects of real world problems that are not reflected in classical models are becoming increasingly important to practitioners: modern optimisation technology provides solutions for many of the original core issues; now the challenge is in those areas that were not considered part of a classical model.

Nevertheless, given the strong solution methods available for many classical problem definitions, it is common, when faced with a new problem, to try and model it as an instance of an existing, classical problem [23]. For example, if a problem can be modelled as a graph colouring problem or even better a linear program, there exist techniques that are very likely to solve the problem adequately. As argued above, however, there are likely to be aspects of the real

* This work was supported by EPSRC research grant GR/M90641 and by Science Foundation Ireland under Grant 00/PI.1/C075.

problem that do not fit easily into a "pure" version of a known combinatorial optimisation problem. The decision must then be made to ignore the problem components that cannot be modelled in a pure way or to add "impure" or customised model components. The drawback of the former approach is that it is not clear that solutions to the problem model are solutions to the real problem. The latter approach also presents challenges, due to the fact that we do not yet understand combinatorial problem solving well enough to be able to predict the effect of adding a customised model component. It can be that what looks like a minor addition to the model significantly changes the performance of standard problem solving techniques.

A case in point are real world scheduling problems. The Artificial Intelligence and Operations Research literature has delivered strong problem solving techniques for a variety of pure problem definitions (e.g., job shop scheduling problem (JSP), resource constraint project scheduling problem). It is rare for a pure JSP to be found in the wild. There are typically additional constraints to deal with like transition times between tasks, multiple (contradictory) optimisation criteria, and resource alternatives. In fact, when faced with a real problem it is sometimes difficult to decide which type of pure problem model to start with: though it may look like a scheduling problem perhaps some characteristics mean that vehicle routing technology may be a better choice. For a complicated problem with some scheduling and some vehicle routing characteristics, we do not yet understand the problem models and problem solving techniques well enough to know which pure model is a better fit.

This paper addresses the relationship between problem characteristics and search performance. In previous work [19], it was shown that scheduling technology performs poorly on reformulated vehicle routing problems while vehicle routing technology performs poorly on reformulated scheduling problems. Here, we investigate the reasons behind these results. We perform an in-depth study by systematically transforming classical vehicle routing problems (VRP) such that they become more like scheduling problems and, symmetrically, by transforming classical scheduling problems so that they look more like VRPs. We then solve these problems with routing and with scheduling technologies, to discover what problem characteristics influence the problem solving techniques.

This paper is organised as follows. We start by defining vehicle routing problems and scheduling problems, and show how we can reformulate VRPs as scheduling problems, and vice versa. We then identify several structural characteristics that we believe differentiate routing problems from scheduling problems. We describe the design of our experiments, to investigate the effect of these problem characteristics on the relative performance of routing and scheduling technology. The results of these experiments are reported and followed by a conclusion.

2 Problem Definitions: VRP & JSP

In the capacitated vehicle routing problem with time windows, m identical vehicles initially located at a depot are to deliver discrete quantities of goods to n customers. Each customer has a demand for goods and each vehicle has a capacity. A vehicle can make only one tour starting at the depot, visiting a subset of customers and returning to the depot. Time windows define an interval for each customer within which the visit must be made. A solution is a set of tours for a subset of vehicles such that all customers are served only once and time window and capacity constraints are respected. The objective is to minimise distance travelled, and sometimes additionally to reduce the number of vehicles used. The problem is NP-hard [14].

An $n \times m$ job shop scheduling problem consists of n jobs and m resources. Each job is a set of m completely ordered activities, where each activity has a duration for which it must execute and a resource which it must execute on. The total ordering defines a set of precedence constraints, meaning that no activity can begin execution until the activity that immediately precedes it in the complete ordering has finished execution. Each of the m activities in a single job requires exclusive use of one of the m resources defined in the problem. No activities that require the same resource can overlap in their execution and once an activity is started it must be executed for its entire duration (i.e. no preemption is allowed). The job shop scheduling decision problem is to decide if all activities can be scheduled, given for each job a release date of 0 and a due date of the desired makespan D, while respecting the resource and precedence constraints. The job shop scheduling decision problem is NP-complete [14].

While not part of the basic JSP definition, transition times and resource alternatives have been reported in the scheduling literature [12]. In the case of a transition time, there is a temporal constraint specifying a minimum time that must expire between pairs of activities executed on the same resource. When alternative resources are represented, each activity has a set of resources that it can be performed on (and this extension of the JSP is known as the flexible job shop problem).

3 Mutual Reformulations

In [5] two reformulations are presented: a reformulation of VRP as a flexible shop problem with setups, and symmetrically a reformulation of JSP as a VRP.

We reformulate the VRP into a scheduling problem as follows. Each vehicle is represented as a resource, and each customer visit as an activity. The distance between a pair of visits corresponds to a transition time between respective activities. Each activity can be performed on any resource, and is constrained to start execution within the time window defined in the original VRP. Each activity has a demand for a secondary resource, corresponding to a visit's demand for capacity of a vehicle. For each resource R there are two special activities $Start_R$ and End_R. Activities $Start_R$ and End_R must be performed on resource R. $Start_R$

must be the first activity performed on R and End_R the last. The transition time between $Start_R$ and any other activity A_i corresponds to the distance between the depot and the i^{th} visit. Similarly the transition time between End_R and A_i corresponds to the distance between the depot and the i^{th} visit. The processing time of $Start_R$ and End_R is zero. We associate a consumable secondary resource with every (primary) resource to model the capacity of vehicles. Consequently a sequence of activities on a resource corresponds to a vehicle's tour in the VRP. In the resultant scheduling problem each job consists of only one activity, each activity can be performed on any resource, and there are transition times between each pair of activities. The problem is then to minimise the sum of transition times on all machines and maybe also to minimise the number of resources used.

We reformulate a scheduling problem into a VRP as follows. We have for each resource a vehicle, and for each activity a customer visit. The visits have a duration the same as that of the corresponding activities. Each visit can be made only by the vehicles corresponding to the set of resources for the activity (more than one resource in the case of flexible JSPs, and only one in the case of pure JSPs). Any ordering between activities in a job results in precedence constraints between visits. Transition times between activities correspond to travel distances between visits. The deadline D imposes time windows on visits. Assuming we have m resources, and therefore m vehicles, we have $2m$ dummy visits corresponding to the departing and returning visits to the depot. A vehicle's tour corresponds to a schedule on a resource. For the $n \times m$ JSP we have a VRP with $m(n+2)$ visits and m vehicles. Since there are no transition times in the JSP, there are no travel distances between visits. There are precedence constraints between those visits corresponding to activities in a job. The decision problem is then to find an ordering of visits on vehicles that respects the precedence constraints and time windows.

4 Problem Characteristics

The performance difference between VRP and scheduling techniques must be due to the characteristics on which the problems differ. In this section, we identify six main characteristics that we believe are sufficient to explain the performance differences between the VRP and scheduling techniques.

Alternative Resources Perhaps the most obvious difference between standard VRP and scheduling problems is the number of alternative resources that may be used for each operation. In vehicle routing there are typically many vehicles that can be used to perform a visit. For example, the standard Solomon benchmarks [21] have twenty-five identical vehicles. In contrast, scheduling problems tend to have very few alternative resources and, as noted, in the pure JSP there are no resource alternatives. This has been reflected in the scheduling literature. For example, Focacci et al. [12] experiment on problems with up to three alternatives while Davenport & Beck [10] use problem instances with at most eight alternatives. We expect few resource alternatives to favour scheduling techniques while many should improve the performance of VRP techniques.

Temporal Constraints In pure VRPs each visit is independent, i.e. there are no constraints requiring a visit to have some temporal relation with other visits. In contrast, scheduling problems typically have long chains or complex directed acyclic graphs of temporal relationships among activities. As a consequence temporal reasoning has been widely explored in the literature [8] and is an important component of many scheduling algorithms [16]. We therefore predict that the performance of scheduling technology should improve as more complex temporal constraints are added to problems.

The Ratio of Operation Duration to Transition Time In pure VRPs a visit has no duration. In contrast, in pure JSP the transition time between operations is zero. The modelling abstraction for scheduling is obviously opposite to the VRP: the transition time between activities on the same resource is so small that it can be ignored. These two problem models are extremes. In both the literature and in the real world there are VRPs with visit durations and scheduling problems with transition times and costs. However, even then the ratio of operation duration to transition time tends to be different. In VRPs the ratio is very small while in scheduling it is often very large. This difference is reflected in solution techniques. Many VRP techniques (e.g., the savings heuristic [9]) look exclusively at transition times when searching, while many of the strong constraint propagation [18, 17] and heuristic search techniques [2, 20] in scheduling ignore transition time. Overall, we expect that the smaller the ratio of operation duration to transition time, the better the VRP techniques should perform relative to the scheduling techniques.

Optimisation Criterion In a VRP the standard optimisation criterion is to minimise the total distance travelled by each vehicle. In scheduling, a common criterion is the minimisation of makespan: the time between the start of the first operation and the end of the last operation. While many criteria have been studied (e.g., tardiness [1], earliness and tardiness [3], flow time, transition times [12]) makespan has received a great deal of attention in the literature even though its relevance has been questioned in practical applications [4, 13]. We expect that the performance of the scheduling techniques should be favoured when using the minimisation of makespan as the criterion.

Temporal Slack Although it seems difficult to make specific predictions as to how temporal slack, i.e. the difference between the time window for an operation and the operation's duration, will effect the performance of the search techniques in both routing and scheduling, we think it worthwhile to experiment with it as well. Slack can be important while solving both routing and scheduling problems: there are state-of-the-art global constraint propagation algorithms and efficient search heuristics based on temporal slack [20].

Resource Capacity Resource capacity in a VRP is the maximum number of visits that a vehicle can perform in any solution. As the resource capacity is decreased, the minimum number of vehicles required in any solution increases. In a scheduling problem, resource capacity is defined as the number of activities that can simultaneously use a resource. These are two different notions that should not be confused. We adopt the VRP definition here.

Reducing resource capacity in a VRP results in more tightly constrained visits since an increasing subset of visits cannot be performed by the same vehicle. We predict that such tighter side constraints will make it relatively more difficult for the VRP techniques to find good solutions as compared with the scheduling techniques which, through constraint propagation, will exploit these constraints better during search. Reducing the VRP definition of resource capacity in a scheduling problem is meaningless for a pure JSP since all activities are assigned a resource. Instead of starting from a pure JSP, the base scheduling problem in this case is a flexible JSP where each activity can be executed on any resource. The reduction in resource capacity has a similar impact as in VRPs: an increasing subset of activities cannot be scheduled on the same resource. However, because we are dealing with flexible JSPs it is hard to make predictions as to the impact of changing resource capacity since the base problem is already impure.

These characteristics may be sufficient to explain the performance differences between VRP and scheduling techniques, because they are sufficient to transform one problem model to the other. Starting with a pure VRP, if we reduce the alternative resources to a singleton for each task, add chains of temporal constraints, reduce the transition time to zero while increasing task duration, and change the optimisation criterion to makespan minimisation, we have a pure JSP problem.

Real Problem Characteristics All of the above are typical of real world problems. For example, in VRPs the vehicle fleet is rarely homogeneous. It is typical that we have classes of vehicles that are specialised for certain tasks, such as tankers or refrigeration lorries, or drivers with specific skills. Precedence constraints are not uncommon in delivery problems, and problems where we route service engineers to tasks [7]. And sometimes we are not concerned about reducing the amount of travel, but in getting all the visits done as quickly as possible.

The same holds true for scheduling problems. It is not exceptional to have a small number of expensive resources that can be reconfigured to perform a variety of very different tasks (i.e. giving a high number of alternative resources for each activity), with high transition costs. In such an environment the objective might be to minimise the amount of time spent reconfiguring machines, thus increasing returns on investment. Therefore, the classical problem model that best corresponds to a real world problem is unclear.

5 Experimental Design

Our goal is to empirically determine which of the above six problem characteristics actually have an effect on the relative performance of scheduling techniques compared to VRP techniques.

We perform two symmetric sets of experiments. In the first set of experiments, we take pure VRP instances and examine the effect of varying one problem characteristic at a time. We generate instances of VRPs. We then solve each problem twice: first we model the problem as a VRP and solve it using routing technology; second, we model the VRP as a scheduling problem (as described in sect. 3) and solve it with scheduling technology. We then measure the relative performance of the technologies. For each problem characteristic, therefore, we begin with a set of pure VRP instances and then generate additional problem sets by varying the parameter in question. Intuitively, as the parameter value deviates more from the VRP, the problems should look increasingly like scheduling problems.

In a symmetric set of experiments, we take pure JSP instances and again vary one problem parameter at a time. We generate pure and transformed JSPs and, as for the VRPs above, apply both scheduling and routing technologies to each instance and measure their relative performance.

5.1 Problems and Their Parameters

Across all problems, the working day starts at midnight and lasts 24 hours. Time is measured in minutes, consequently the origin of our schedule is 0 and the horizon is 1440. Each operation duration is 20 minutes. Each problem set in our experiments has 100 instances.

VRP instances Every VRP instance has 1 depot, 25 vehicles, 100 customers. Distance is measured in decameters using the Manhattan metric. The other parameters for generating VRPs are listed below.

JSP instances Each of the instances has 10 machines and 10 jobs, i.e., 10 totally ordered sets of operations. In every instance of the initial set, a single machine is specified for each operation. There are no transition times between operations on a given machine. The other parameters for generating JSPs are displayed below.

n the number of operations:
  VRP: the number of customers. In all problems $n = 100$.
  JSP: the number of activities. In all problems $n = 100$.
m the number of resources:
  VRP: the number of vehicles. In all problems $m = 25$.
  JSP: the number of machines. In all problems $m = 10$.

p resource specialisation. The parameter p represents the proportion of the pool of resources (vehicles or machines) that can perform each operation. For operation i, the number of possible resources m_i is calculated as:

$$m_i = \lceil p \times m \rceil \tag{1}$$

Small values of p correspond to few resource alternatives and therefore correspond more to scheduling problems.

VRP: the default value is $p = 1.0$ (any vehicle can do any visit).

JSP: the default value is $p = 0.1$ (each operation has a single specified machine).

pc The (integer) number of precedence constraints. This parameter specifies the number of pairs of visits whose sequence is constrained such that visit i must end before the start of visit j. Note that i and j are not necessarily on the same resource. Large values of pc correspond to many precedence constraints characteristic of scheduling problems.

VRP: the default value is $pc = 0$.

JSP: the default value is $pc = 450$ (10 jobs with 10 operations each).

ρ The ratio of visit duration to travel time for the VRP or the ratio of activity duration to transition time for the JSP. This parameter is used to set V, the speed of the vehicles or the speed of transitions on machines:

$$V = \frac{\rho \times \overline{d}}{\overline{\tau}} \tag{2}$$

where $\overline{\tau}$ is the average duration of visits (activities), and $\overline{d}$ is the average distance (transition time) between them. Large ρ values are typical of scheduling problems while small ρ values correspond to VRPs.

VRP: the default value is $\rho = 1.0$.

JSP: the default value is $\rho = \infty$ (transition time equals zero).

σ Normalised slack.

$$\sigma = \frac{le_i - \tau_i - es_i}{le_i - es_i}, \tag{3}$$

where τ_i is the duration of operation i, and le_i and es_i are the latest end and earliest start of operation i. It follows from this definition that $\sigma \in (0, 1)$ and that the larger σ becomes, the wider the time window is for the same τ_i.

VRP: the default value is $\rho = 0.9$.

JSP: the default value is $\rho = 0.986$ (time windows are as long as the working day).

c The optimisation criterion:

VRP: the default value is $c =$ *minimise total travel time.*

JSP: the default value is $c =$ *minimise makespan.*

R_c Average resource capacity:

$$R_c = \frac{\Sigma_{i=1}^{m} r_{c_i}}{n} \tag{4}$$

where r_{c_i} is the capacity of vehicle (machine) i measured as the number of operations (customer visits or activities) that resource i can perform, m is the number of resources, n is the overall number of operations.

VRP: the default value is $R_c = 25.0$ (r_{c_i} is set constant such that each one of $m = 25$ vehicles can perform $n = 100$ visits).

JSP: the default value is $R_c = 10.0$ (r_{c_i} is set constant such that each one of $m = 10$ machines can perform $n = 100$ operations).

5.2 Problem Generation

Pure VRP and JSP problems are generated with the default parameters, specified above. All generated problems [24] are tested for solubility, and are rejected if not found to be soluble within 5 minutes of CPU time.

Generating VRPs To create the initial set of pure VRP instances, the geographic coordinates of each visit are randomly drawn with replacement from a data file containing all postal codes of locations in the city of Glasgow, within a 5 km radius of the city centre. These postal codes can be directly translated to coordinates with an accuracy of ten meters. Because we are using city coordinates, all distances are Manhattan distance.

We have also generated additional problem sets varying one parameter at a time while all the remaining parameters take their default values: $p = 1.0$, $R_c = 25.0$, $pc = 0$, $\rho = 1.0$, $\sigma = 0.9$, $c =$ *minimise total travel time*.

p Resource specialisation is varied such that $p \in \{0.05, 0.1, 0.25, 0.5, 1.0\}$. With $m = 25$ this corresponds to VRPs with 1, 3, 6, 13, and 25 possible vehicles per visit respectively. For a given p to get the corresponding numbers of vehicle alternatives, we take a visit and randomly select without replacement the appropriate number of vehicles.

R_c Vehicle capacities take on the following values: $R_c \in \{1.25, 2.50, 6.25, 25.00\}$. This corresponds to resources capable of performing 5, 10, 25 and 100 customer visits.

ρ The visit duration to travel time ratios are as follows: $\rho \in \{0.25, 0.5, 1.0, 2.0, 2.25, 3.0\}$. All visit durations were set at 20 minutes, consequently problems have a vehicle speed ranging from 12km/h to 72km/h.

pc The different numbers of precedence constraints are $\{0, 50, 450, 1200\}$. Each precedence constraint imposes an ordering for two given visits, without forcing these two visits to use the same vehicle. This corresponds to varying numbers and lengths of totally ordered precedence chains such that 5 chains each involving 5 visits are equivalent to a pc of 50, 10 chains with 10 visits each to a pc of 450, 4 chains with 25 visits each to a pc of 1200.

c In the 100 pure VRPs we change the optimisation criterion from *total travel time* to *makespan*.

σ Normalised temporal slack $\sigma \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. As σ increases, time windows get larger with respect to visit durations. Given σ, visit duration, the start and end of the working day, we randomly choose earliest start times of visits and compute latest start times using (3).

Generating JSPs Each JSP generated is a 10×10 intance. The default parameters of generated JSPs are given above.

We have also generated additional problem sets varying one parameter at a time. Unless otherwise stated, all the remaining parameters take their default values: $p = 0.1$, $R_c = 10.0$, $pc = 450$, $\rho = \infty$, $\sigma = 0.986$, $c =$ *minimise makespan*.

p Resource specialisation $p \in \{0.1, 0.25, 0.5, 1.0\}$. This corresponds to each operation having 1, 3, 5 and 10 alternative machines. Instance generation is analogous to VRPs.

R_c Because varying resource capacity in pure JSPs does not have an influence on the quality of solutions, we experiment with flexible job shop instances. We generate additional problem sets such that in each 10×10 instance every activity has 10 alternative resources ($p = 1.0$). All the remaining parameters take their default values as above. Resource capacity varies such that $R_c \in \{1.3, 2.5, 6.3, 10.0\}$. This corresponds to each machine being capable of performing 13, 25, 63 and 100 activities.

ρ Activity duration to transition times ratio is chosen as follows: $\rho \in \{0.5, 1.0, 3.0, 10.0, \infty\}$. A ρ of ∞ corresponds to pure JSPs with infinitely fast transitions between operations on a machine.

pc We vary the number of precedence constraints such that $pc \in \{0, 40, 126, 450\}$. With respect to pc in isolation, removing precedence constraints correspond to making the instances more like the VRP.

c In the 100 pure JSPs we change the optimisation criterion from *minimise makespan* to *minimise total processing time*. Total processing time minimisation can be thought of as minimising the sum of makespans over all machines.

σ Normalised temporal slack $\sigma \in \{0.65, 0.75, 0.85, 0.95, 0.986\}$. To obtain soluble JSPs with varying σ's, we first impose the necessary number of precedence constraints and then compute time windows for activities.

Problem Specification Our VRP-JSP problem instances have a unified format that allows us to represent the whole spectrum of problems with varied parameters across the VRP and JSP domains. The specification is displayed below.

```
1.  # instance filename
2.  # generation parameters (random generator seeds etc.)
    ...
```

```
3.  OptCriterion
4.  n m
5.  V
6.  Origin Horizon
7.  XDepot YDepot
8.  id Dur Demand PostCode1 PostCode2 EST LFT X Y res1 res2 .. -1
    ...
9.  0    TT01 ... TT0n
    ...  ...  ... ...
    TT10 0    ... TTin
    ...  ...  ... ...
    TTn0 TTn1 ... TTnn
10. resourceID Capacity
11. ...
12. Diff nDiff
    ...
13. Same nSame
    ...
14. Precedence nPrecedence
    ...
```

The specification has several sections. In the first section, comments, each line starts with a hash symbol ('#'). Here we record the instance name and problem generator parameters that were used to produce an instance.

In the general information section (lines 3 - 7) we specify the optimisation criterion, the number of resources n and the number of operations m. Parameter V (line 5) is the speed of vehicles for the VRPs and speed of transitions between activities in the JSPs. It is computed using (2). Line 6 specifies the start and end time of the working day, line 7 the x,y-coordinates of the depot in the VRPs (0's for the JSPs).

The operations section (starting with line 8) describes each operation in a separate line as follows: visit ID, duration in minutes, demand for resource capacity (in all our instances it equals 1), alphanumeric postal codes of the respective customer (if it is a JSP, we ignore it), earliest start and latest finish times, x,y-coordinates of the customer (0's for JSPs) and finally the list of resources allowed to perform this operation followed by '- 1'.

In the transition time matrix section (starting with line 9) we explicitly display the $n+1 \times n+1$ transition time matrix with the first row and column corresponding to the depot. Clearly, for the pure JSPs every element of this matrix is zero. Even though an explicit transition time matrix is redundant for pure VRPs because we can always compute distances having coordinates of locations; we need it to be able to represent non-pure VRPs with transition times independent of distance as well as JSPs with (zero) transition times.

In the resources section (line 10) we list all the resources with their capacities measured as the number of operations they can perform.

Finally, lines 12 - 14 belong to the side constraints section. Each side constraint occupies a separate line and involves a pair of operations $\{o_i, o_j\}$ and states that the operations must be performed by different resources ($Diff$ side constraints), by the same resource (*Same* constraints) or says that the operation o_i should be completed before o_j starts on the same or different resource. $nDiff$, *nSame* and *nPrecedence* show how many constraints of each type we have.

5.3 Solution Technology

In our experiments we use the ILOG optimisation suite. The scheduling library (ILOG Scheduler 5.2) has global constraint propagation [1, 16, 18] within constructive tree search as its core technology, while the vehicle routing library (ILOG Dispatcher 3.2) focuses on local search [11, 15].

As part of the routing technique, we apply first accept neighbourhood search enhanced by the guided local search meta-heuristic [22] with a penalty factor of 0.4. To construct a neighbourhood of a solution, we use the standard move operators (*TwoOpt*, *OrOpt*, *Relocate*, *Cross* and *Exchange*). Search starts from the first solution found by the savings heuristic or uses the first solution found by the scheduling technique. The scheduling technique is constructive depth first search with standard slack based heuristics [20], global constraint propagation, and edge finding [1].

5.4 Evaluation Criteria

Our primary evaluation criterion is λ, the ratio of the cost of the best solution found by the scheduling technique to that found by the VRP technique in a given CPU time limit. Specifically:

$$\lambda = \frac{C_{sched}}{C_{rout}}, \tag{5}$$

where C_{sched} and C_{rout} are the cost of the best solutions found by the scheduling and routing techniques in the given amount of CPU time. Because we are minimising, routing technology is outperforming scheduling technology when $\lambda > 1$. When $\lambda < 1$, scheduling technology dominates. And, finally, when $\lambda = 1$ the technologies perform equally.

Apart from λ, we also check if technologies are capable of finding solutions within a specified time limit.

5.5 Choosing the CPU time limit

To choose the CPU time bound for our empirical study we ran preliminary experiments. In each experiment we randomly took 20 instances out of 100-problem sets of VRPs with varying operation duration to travel time ratios ρ. Each problem was solved using different CPU time limits up to 50 minutes as displayed in Fig. 1. Each point in Fig. 1 is a mean λ over 20 instances test problems displayed against ρ and CPU time limit. We see that there is no significant change of λ after 10 minutes. An analysis of the results shows that neither solution technology make significant improvements in solution quality. As a result, the CPU time bound is chosen to be 10 minutes.

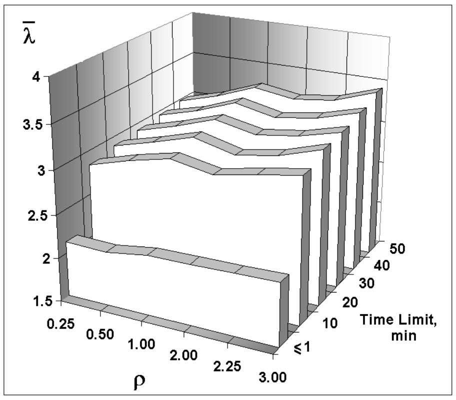

Fig. 1. Influence of CPU time on λ for different values of ρ. Each value of λ is an average of 20 VRPs with $p = 1.0$, $pc = 0$, $R_c = 25$, $\sigma = 0.9$ and $c = \textit{minimise total travel time}$.

6 The Empirical Results

6.1 Base cases: pure VRPs and JSPs

In this section we present the experimental results obtained by scheduling and routing technologies on two base problem sets, pure VRPs and pure JSPs.

In Fig. 2 we display two scatters of λ, one for the base VRP instances and the other for the base JSP instances. Clearly, the routing technology dominates for the pure VRP instances ($\lambda > 1$) and the scheduling technology dominates for the

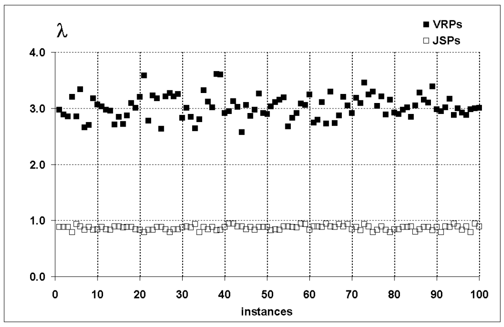

Fig. 2. Base case: λ's for 100 pure JSPs and 100 pure VRPs. The default parameters for the VRPs are $p = 1.0$, $\rho = 1.0$, $pc = 0$, $R_c = 25$, $\sigma = 0.9$, $c = \textit{min travel time}$. The default parameters for the JSPs are $p = 0.1$, $\rho = \infty$, $pc = 450$, $R_c = 10.0$, $\sigma = 0.986$, $c = \textit{min makespan}$; search starts from common initial solutions found by the scheduling technique.

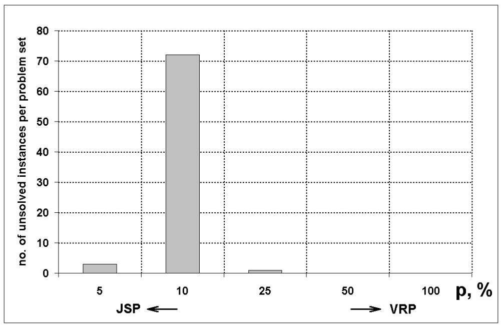

Fig. 3. The number of instances per problem set that could not be solved by the routing technique as we increase the number of alternative resources in an instance.

pure JSPs ($\lambda < 1$). In the experiment with pure JSPs the routing technology was not able to come up with a first solution independently because the first solution construction heuristics are sensitive to impurities in the VRPs such as resource allocation constraints and temporal relationships between activities [6]. Therefore in this and all subsequent experiments with JSPs both technologies started search from common first solutions found by the scheduling technology.

We now study the influence of the following: variation in the number of alternative resources, resource capacity, the number of precedence constraints, the influence of operation duration versus transition times, optimisation criteria, and finally slack.

6.2 Alternative Resources (p)

VRP instances To investigate the influence of resource alternatives on the performance of the solving technologies, we ran a series of experiments using instances with varying p that were generated as explained in sect. 5.2. It turns out that the routing technology could not solve some of the instances with small numbers of alternative resources. Fig. 3 shows how many instances per problem set could not be solved by the routing technology. We observe an insolubility peak at $p = 0.1$. We conjecture that at $p = 0.1$ problems are critically constrained such that there are very few feasible vehicle assignments. Since savings is a single pass heuristic, it does no search for such assignments and therefore fails when they are difficult to construct. With $p < 0.1$, no vehicle choices are necessary since only one vehicle can perform each visit. At $p > 0.1$, many vehicles are possible, increasing the number of feasible vehicle assignments.

In Fig. 4 we present λ with respect to p, expressed as a percentage. Two contours are presented, one for λ for first solution found (λ_{first}) by both techniques, and λ for the best solution found (λ_{best}) by both techniques. Both contours exclude any instance for which the routing technology could not construct a first

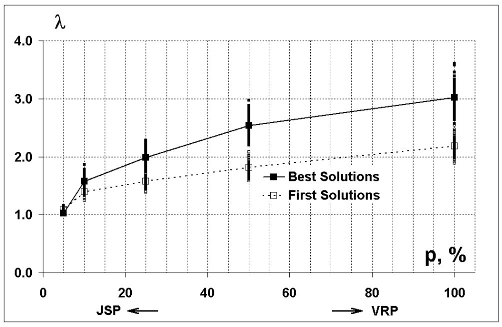

Fig. 4. VRPs: Effects of varying vehicle specialisation. p varies, $pc = 0$, $\rho = 1.0$, $\sigma = 0.9$, $R_c = 25.0$, $c = min\ travel\ time$. Search starts from independently found first solutions.

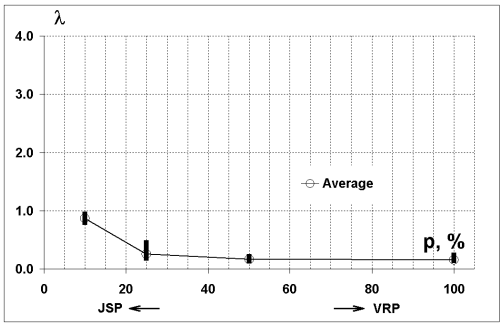

Fig. 5. JSPs: Effects of varying machine specialisation. p varies, $pc = 450$, $\rho = \infty$, $\sigma = 0.986$, $R_c = 10.0$, $c = min\ makespan$. Search starts from common first scheduling solutions.

solution. We see that as the number of vehicles (machines) per visit (activity) increases, VRP technology improves relative to scheduling technology.

JSP instances In Fig. 5 we show how λ depends on the number of alternative resources allowed for an operation. We could expect that as p grows and, consequently, problems become more VRP-like, the routing technology should be favoured. However, we see the opposite effect. The JSP turns out to be not so amenable to the routing technology because, on the one hand, there are many temporal constraints leading to the rejection of many moves and, on the other, a solution neighbourhood for the problem instances of this size is very large.

6.3 Resource Capacity (R_c)

VRP instances To study the influence of resource capacity on search performance we use problem instances with varying resource capacities generated as

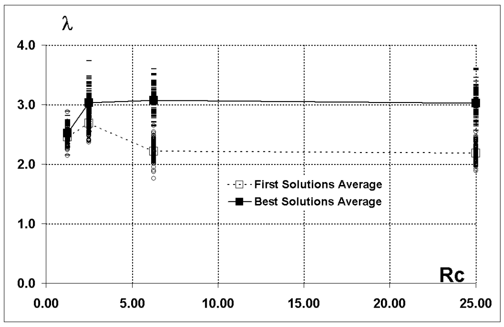

Fig. 6. VRPs: What happens when we vary vehicle capacity? R_c varies, $p = 1.0$, $pc = 0$, $\rho = 1.0$, $\sigma = 0.9$, $c = \mathit{min\ travel\ time}$. Search starts from independently found first solutions.

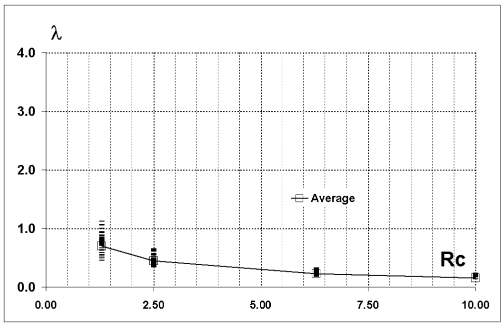

Fig. 7. JSPs: What happens when we vary machine capacity? R_c varies, $p = 0.1$, $pc = 450$, $\rho = \infty$, $\sigma = 0.986$, $c = \mathit{min\ makespan}$. Search starts from common first scheduling solutions.

explained in sect. 5.2. Fig. 6 shows[3] that across the range of varied resource capacity the scheduling technique is dominated by the routing technique. As we increase resource capacity the performance of the scheduling technique becomes relatively worse. This is an expected result because as we have seen the scheduling technology benefits from using global constraints. Consequently, with tighter capacity constraints the scheduling technique performs relatively better than it does with looser constraints.

JSP instances Experimenting with varying resource capacity in pure JSPs is not meaningful because changing machine capacity does not have an influence on the quality of solutions. Consequently we now experiment with flexible job shop instances, i.e. instances where an activity can be performed by any machine.

[3] Recall that varying resource capacity (slack) does not necessarily make a problem more or less like a scheduling problem or a VRP.

Fig. 7 shows that as we increase resource capacity the routing technology is dominated by the scheduling technology. The picture is analogous to the results of the experiments with alternative resources (sect. 6.2). This can be explained by the fact that as we increase capacity, the size of solution neighbourhoods grows. As a consequence, the routing technology cannot efficiently improve solutions as many moves get discarded in the presence of precedence constraints between operations. Note that the routing technology is unable to construct first solutions and therefore it uses those found by the scheduling technique.

6.4 Precedence Constraints (pc)

VRP instances To study the influence of precedence constraints we vary pc. Experiments show that as we introduce precedence constraints the VRP technology fails to produce initial solutions, but the scheduling technology appears to

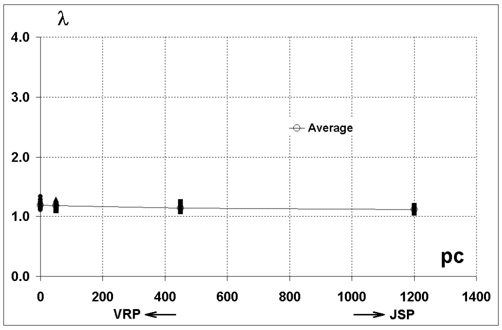

Fig. 8. VRPs: Influence of the number of precedence constraints. pc varies, $p = 1.0$, $R_c = 25.0$, $\rho = 1.0$, $\sigma = 0.9$, $c = min\ travel\ time$. Search starts from common initial scheduling solutions.

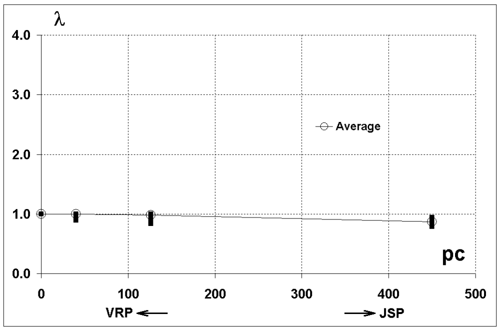

Fig. 9. JSPs: Influence of the number of precedence constraints. pc varies, $p = 0.1$, $R_c = 10.0$, $\rho = \infty$, $\sigma = 0.986$, $c = min\ makespan$. Search starts from common initial scheduling solutions.

be robust. If both techniques start with a scheduling solution (Fig. 8), the VRP technology dominates the scheduling technology (when $pc = 0$, $\lambda \approx 1.2$) but this becomes less significant as we increase the number of precedence constraints (when $pc = 1200$, $\lambda \approx 1.1$).

JSP instances In Fig. 9 we display how relative solution quality λ is affected by the number of precedence constraints in JSPs. We see that as we remove precedence constraints and both technologies start from common initial scheduling solutions, they tend to perform equally ($\lambda \approx 1$). It is an expected result because if we remove *all* precedences from a JSP, the problem becomes tractable. Indeed, subject to no precedence constraints, no alternative resources and zero transition times, we can construct an unsorted list of n operations on each of the m resources in $O(nm)$ time. The length D of the longest list of operations on some resource r_i gives us the makespan. It is optimal because when we have zero transition times D equals the sum of durations of operations on r_i and these operations cannot be relocated to any other resource. Consequently, when $pc = 0$, the first solution found by the scheduling technique is an optimal solution for any instance.

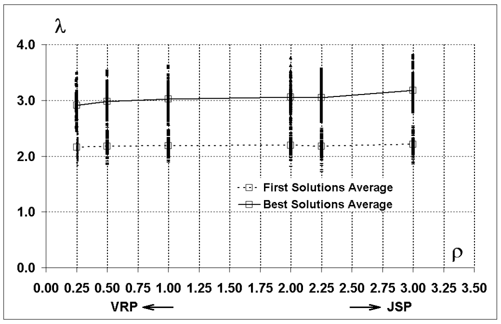

Fig. 10. VRPs: Effects of the ratio of operation duration to transition time. ρ varies, $pc = 0$, $p = 1.0$, $R_c = 25.0$, $\sigma = 0.9$, $c = min\ travel\ time$. Search starts from independently found initial solutions.

6.5 Operation Duration vs Transition Time (ρ)

VRP instances In [5] it was argued that higher ρ's will tend to favour scheduling techniques because large values of ρ correspond to small transition times compared to operation durations, typical of scheduling problems. Our results demonstrate, however, that in isolation this parameter does not bring about such a performance difference and routing technology continues to outperform the scheduling technique (when $\rho = 0.25$, $\lambda \approx 2.9$ and, when $\rho = 3.0$, $\lambda \approx 3.2$, Fig. 10). This is not what we expected.

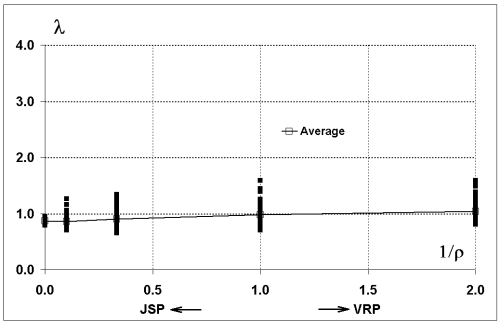

Fig. 11. JSPs: Effects of the ratio of operation duration to transition time. ρ varies, $pc = 450$, $p = 0.1$, $R_c = 10.0$, $\sigma = 0.986$, $c = min\ makespan$. Search starts from common initial scheduling solutions.

JSP instances In Fig. 11 we observe that increasing ρ (i.e. decreasing $1/\rho$ in the picture) for the JSPs marginally improves the relative performance of the routing technique (when $\rho = \infty$ and $1/\rho = 0$, $\lambda \approx 0.87$ whereas when $\rho = 0.5$, $\lambda \approx 1.04$).

6.6 Optimisation Criterion (c)

VRP instances We now solve pure VRPs using two different optimisation criteria: minimise total travel (i.e. from a scheduling perspective, minimise the sum of setups), and minimise makespan (complete all the visits as early as possible). In Fig. 12 we have a scatter plot, on the left λ_{best} and λ_{first} using the minimisation of travel, and on the right λ_{best} and λ_{first} where the criterion is minimising makespan. We see a large difference in the behaviour of the solving technologies. When minimising travel (scatter on the left) the routing technology is consistently 2 to 3 times better than the scheduling technology. When we switch to minimising makespan (scatter on the right) the scheduling and routing technologies perform equally (i.e. λ_{best} and λ_{first} are around 1). These results are particularly dramatic given that all the other parameters are that of a standard VRP: the only difference is the optimisation criterion.

Why is there such a difference when we change optimisation criterion? One answer might be the underlying techniques used in the scheduling technology. When minimising a maximum function (such as makespan), the upper bound can be directly propagated on the completion time of each operation/visit, and the domains of possible start times can be effectively tightened. However, when a sum function (e.g., sum of transition times) is being optimised, the respective constraint has to be considered at each stage of search simultaneously with resource constraints [1] thus making it harder to achieve large domain reductions. The scheduling technology we used is built to exploit the powerful constraint propagation that is possible using such criterion as makespan, whereas the rout-

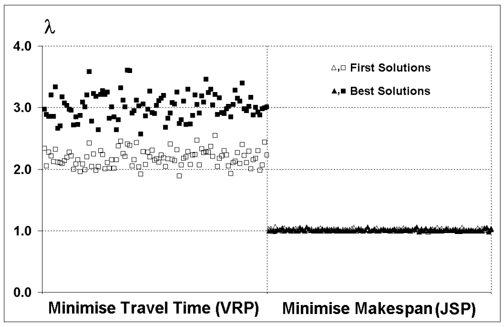

Fig. 12. VRPs: What happens when we swap optimisation for pure VRPs? On the left, *minimise travel time*, and on the right *minimise makespan*. Search starts from independently found first solutions.

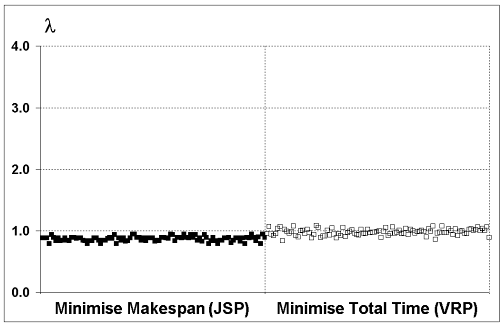

Fig. 13. JSPs: What happens when we swap optimisation for pure JSPs? On the left, *minimise makespan*, and on the right *minimise total time*. Search starts from common first scheduling solutions.

ing technology is not. We might therefore expect when we have a VRP, and the goal is to complete all visits as soon as possible, that we might be better solving it as a scheduling problem.

JSP instances We now experiment with two different optimisation criteria for pure JSPs, makespan and total processing time. In Fig. 13 we have a scatter plot with the minimisation of the makespan on the left and the minimisation of total processing time on the right. When minimising makespan (scatter on the left) the routing technology is consistently dominated by the scheduling technology (the mean λ is 0.868). When we switch to minimising total time (scatter on the right) the routing technology starts to compete with the scheduling technology (the mean λ is 0.977).

These results again demonstrate the importance of the optimisation criterion for the performance of the routing and scheduling techniques. Note that even though the values of λ have not changed significantly for the JSPs, the change in favour of the routing technique does occur in the area of zero transition cost (i.e., where arcs in the routing graph have a zero cost)! Because the cost function calculation in the routing technique is entirely arc-based, we expect this change to be even greater for mixed JSPs with non-zero transition costs.

6.7 Temporal slack (σ)

VRP instances Experiments (Fig. 14) demonstrate that, as slack increases, routing technology improves relative to scheduling technology (from a λ of approximately 2.7 when $\sigma = 0.5$ to a λ of 3.0 when $\sigma = 0.9$). Conversely, when slack decreases it appears that scheduling technology is improving, yet continues to be dominated by the routing technology. We might conclude from this that as slack decreases, and problems become more temporally constrained, we might incorporate scheduling technology into the routing technology.

JSP instances It turns out that tightening slack for JSPs leads to a λ of 1.0 (when $\sigma = 0.986$, $\lambda \approx 0.87$ and, when $\sigma = 0.65$, $\lambda \approx 1.0$ as displayed in Fig. 15). This is because the first solution found by the scheduling technique tends to be optimal as we are making the problems more temporally constrained.

6.8 Summary of Results

We have varied the number of alternative resources, the number of precedence constraints, the operation duration to transition time ratio, the optimisation criterion, and the temporal slack. Each parameter was varied in isolation, in two symmetric sets of experiments. In the first series of experiments, we started from pure VRPs and transformed them systematically so that they became more like JSPs. In the second series, we started from pure JSPs and moved toward VRPs.

We have also varied resource capacity. However, with this parameter to have a meaningful comparison of technique performance, we used pure VRPs in the first set of experiments and flexible job shop in the second.

We can transform a JSP into a VRP or vice versa by simultaneously varying the number of alternative resources, the number of temporal constraints between operations, the optimisation criterion and operation duration vs. transition time. Varying slack or resource capacity does not contribute to this problem transformation.

Fig. 16 pictorially summarises our study. The pure VRP and pure JSP domains are shown as circles. Problem transformations are depicted by means of arrows labelled by the corresponding problem parameters. We also show the response to each parameter change in the relative quality λ of the solving techniques.

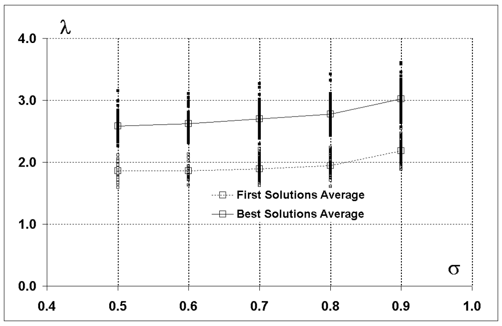

Fig. 14. VRPs: Varying temporal slack. σ varies, $pc = 0$, $p = 1.0$, $\rho = 1.0$, $R_c = 25.0$, $c =$ *min travel time*. Search starts from independently found initial solutions.

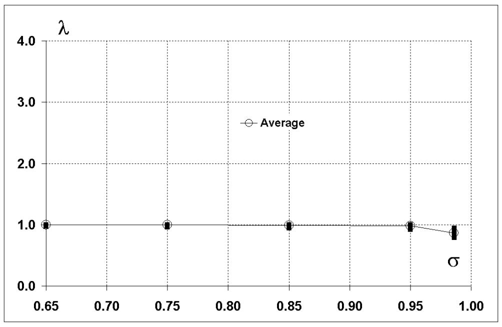

Fig. 15. JSPs: Varying temporal slack. σ varies, $pc = 450$, $p = 0.1$, $R_c = 10.0$, $\rho = \infty$, $c =$ *min makespan*. Search starts from common initial scheduling solutions.

The most prominent effects on the performance of the solving techniques were brought about by varying:

the number of alternative resources p The removal of alternative vehicles in the VRPs favours the scheduling technology but the addition of alternative machines to the JSPs proves even less amenable to the routing technique than pure JSPs;

resource capacity R_c The effects of varying resource capacity are analogous to the effects of changing the number of alternative resources. Decreasing capacity in VRPs favours the scheduling technology although it remains dominated by the routing technique. In contrast, increasing resource capacity in flexible JSPs leads to even more pronounced dominance of the scheduling technique.

the number of temporal constraints between activities pc An increase in pc in VRPs favours the scheduling technique, and, symmetrically, a decrease in pc in the JSP improves the performance of the routing technique;

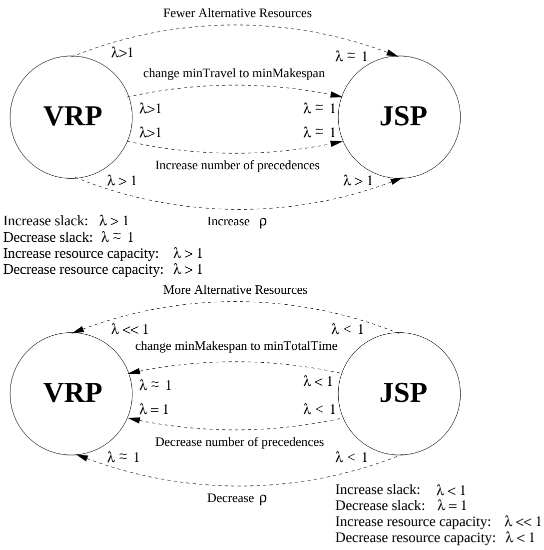

Fig. 16. Influence of various problem characteristics on the performance of VRP and JSP technologies. Arrows represent reformulation direction, arrow labels show the varied parameter. $\lambda > 1$ means routing technology outperforms scheduling technology, when $\lambda < 1$ scheduling technology is a winner, $\lambda \approx 1$ shows that both technologies perform equally.

optimisation criterion c Changing c from total travel minimisation to makespan minimisation in VRPs notably favours the scheduling technology. Symmetrically, changing c from makespan minimisation to total processing time minimisation in JSPs favours the routing technique.

7 Conclusions

We have considered the vehicle routing and job shop scheduling problems and carried out an empirical study to develop an understanding of how their characteristics contribute to the performance of standard solving technologies. We have identified six such parameters: vehicle specialisation, complex temporal relationships between operations, the optimisation criterion, operation duration to travel time ratio, resource capacity and temporal slack. In the experiments, we started from pure VRPs and, separately, from pure JSPs and varied those parameters, each in isolation, to assess their influence on the performance of the solving techniques.

In our study we used VRP and scheduling technology found in commercially available tools. This may appear naive. However, we believe that this is a crucial feature of our study. The current research literature boasts a number of powerful approaches for solving combinatorial optimisation problems (e.g., constraint programming, mixed integer programming, SAT solvers, etc.). For problems that do not fall directly into a classical model, however, these solution techniques require significant skill and experience to apply to their full potential. In this study, we were not interested in the algorithmic customisations we could make to scheduling or VRP technology to allow it to solve "impure" problems better. Rather we were interested in changing the problem characteristics of classical models and studying the impact on existing solution techniques. Understanding how changes in classical models effect solution techniques is critical in extending the scope of optimisation technology to more realistic problems. Furthermore, one of our long term goals is to decrease the expertise required to customise algorithms for impure problems. We plan to investigate the automation of the process of problem analysis to be able to suggest appropriate solution technology. The identification of the relationship between problem characteristics and search performance is an important step in this direction.

References

1. Ph. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems.* Kluwer Academic Publishers, 2001.
2. J. C. Beck and M. S. Fox. Dynamic problem structure analysis as a basis for constraint-directed scheduling heuristics. *Artificial Intelligence*, 117(1):31–81, 2000.
3. J. C. Beck and P. Refalo. A hybrid approach to scheduling with earliness and tardiness costs. *Annals of Operations Research*, 2002. in press.
4. J.C. Beck, A.J Davenport, and M.S. Fox. Five pitfalls of empirical scheduling research. In *3rd Int. Conference on Principles and Practice of Constraint Programming (CP'97)*, 1997.
5. J.C. Beck, P. Prosser, and E. Selensky. On the reformulation of vehicle routing problems and scheduling problems. In *LNAI 2371, Proceedings of the Symposium on Abstraction, Reformulation and Approximation (SARA 2002)*, pages 282–289, 2002.
6. J.C. Beck, P. Prosser, and E. Selensky. Vehicle routing and job shop scheduling: What's the difference? In *Proceedings of the 13th International Conference on Artificial Intelligence Planning and Scheduling*, 2003.
7. C. Brind, C. Muller, and P. Prosser. Stochastic techniques for resource management. *BT Technology Journal*, 13:55–65, 1995.
8. A. Cesta, A. Oddi, and S.F. Smith. A constraint-based method for project scheduling with time windows. *Journal of Heuristics*, 8(1):109–136, 2000.
9. G. Clarke and J.W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 46:93–100, 1964.
10. A.J. Davenport and J.C. Beck. An investigation into two approaches for constraint directed resource allocation and scheduling. In *INFORMS*, 1999.
11. B. DeBacker, V. Furnon, P. Shaw, P. Kilby, and P. Prosser. Solving vehicle routing problems using constraint programming and metaheuritics. *Journal of Heuristics*, 6:5001–523, 2000.
12. F. Focacci, P. Laborie, and W. Nuijten. Solving scheduling problems with setup times and alternative resources. In *Proceedings of the 5th International Conference on Artificial Intelligence Planning and Scheduling*, 2000.

13. M.S. Fox. *Constraint-Directed Search: A Case Study of Job Shop Scheduling*. PhD thesis, Carnegy Mellon University, Intelligent Systems Laboratory, The Robotics Institute, Pittsburgh, PA, 1983.
14. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.
15. P. Kilby, P. Prosser, and P. Shaw. A comparison of traditional and constraint-based heuristics methods for vehicle routing problems with side constraints. *Constraints*, 5(4):389–414, 2000.
16. P. Laborie. Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. In *Proceedings of the 6th European Conference on Planning (ECP01)*, 2001.
17. C. Le Pape. Implementation of Resource Constraints in ILOG SCHEDULE: A Library for the Development of Constraint-Based Scheduling Systems. *Intelligent Systems Engineering*, 3(2):55–66, 1994.
18. W. P. M. Nuijten. *Time and resource constrained scheduling: a constraint satisfaction approach*. PhD thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1994.
19. E. Selensky. On mutual reformulation of shop scheduling and vehicle routing. In *Proceedings of the 20th UK PLANSIG*, 2001.
20. S. Smith and C. Cheng. Slack based heuristics for constraint satisfaction scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 139–144, 1993.
21. M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Operations Research*, 35:254–365, 1987.
22. C. Voudouris and E.P.K. Tsang. Guided Local Search. *European Journal of Operational Research*, 113(2):80–110, 1998.
23. T. Walsh. Reformulating propositional satisfiability as constraint satisfaction. In *Symposium on Abstraction, Reformulation and Approximation (SARA)*, 2000.
24. *PRAS* vehicle routing and shop scheduling problem instances. http://www.dcs.gla.ac.uk/pras/resources/.