Scheduling in a Virtual Enterprise in the Service Sector

Florian Kandler

Electronic Commerce Competence Center, Donau-City-Strasse 1, 1070 Vienna, Austria
florian.kandler@ec3,at
http://www.ec3.at/

Abstract. The work described in this paper presents an approach on scheduling in a virtual enterprise in the tourism sector. This scheduling scenario has to reconcile customers' and service providers' interests. That is, the customer's temporal constraints on when to consume the service, and the service provider's preferences on when to accept a booking. An algorithm is being developed for this situation, that incorporates both sides' interests by first taking the customers restrictions, and then finding the highest preferred solution for the service provider within these limits. This is not done in a setting of competing agents each solving his local problem [1], but in a single composite problem that is solved by the virtual enterprise using the algorithm presented in this paper.
The research described in this paper is part of the MOVE (Management and Optimization of Business Processes in Virtual Enterprises) Project of the Electronic Commerce Competence Center in Austria, whose ultimate goal is to develop a reference implementation of a virtual enterprise in the tourism sector, where the field of tourism is just one scenario where we want to try and prove our ideas and findings for the more general field of virtual enterprises in the service sector.

1 Introduction

The work presented here is part of the development of a virtual enterprise system in the tourism sector [3]. As a whole system, the virtual enterprise shall enable the distributed community of service providers in the tourism sector to jointly achieve higher sales by using internet technology. Therefore, the disperse scheduling that happens at every service provider locally for bookings that are received via traditional means like phone and fax is integrated into the virtual enterprise system to support ad hoc information about available services and booking for the customer. The virtual enterprise will support the customer in assembling his individual holiday package by offering advice and suggesting services, based on the customer's profile and previous bookings, as well as helping him to find the best and cheapest offer. The work presented in this paper logically follows this process of holiday package definition, taking those decision as given, and suggesting a scheduling algorithm that reconciles these customer wishes with the preferences of the service providers with respect to their current booking situation.

All together, the presented algorithm that is used in the virtual enterprise is part of the automated scheduler, that enables the system to offer the customer the swift replies needed in the e-marketplace. On the other hand, this algorithm plays a central role in supporting the service providers to assert their preferences.

2 Problem description

This paper will examine the scheduling problem in a virtual enterprise in the tourism sector. The basic functionality of the virtual enterprise to the customer shall be to offer a one-face-to-the-customer service. That is, the customer can browse through a collection of all the offered services of a large set of service providers that participate in the virtual enterprise. Above that, the virtual enterprise will support the customer in finding the best and cheapest offers.

To the service provider, the virtual enterprise offers a matching and sales functionality, e.g. the virtual enterprise routes customers' service requests to the appropriate service providers. Furthermore, the virtual enterprise actively endorses service providers' interests and preferences, e.g. preferred booking times, preferred customers, etc. and supplies the service providers with aggregate information of customers' demands.

So, the virtual enterprise takes a reconciliation function between the interests of customers and service providers. This has to be incorporated into the scheduling algorithms.

3 Scheduling in a Virtual Enterprise

3.1 Basic scheduling procedure in the Virtual Enterprise

As outlined above, the Virtual Enterprise in our tourism scenario shall offer a one-face-to-the-customer approach that enables the customer to assemble a complete holiday package. That is, an arbitrary composition of different kinds of services will be assembled by the customer. The overall problem will break down into two layers of scheduling problems. On the upper layer, the scheduler will have to find a sequence of the selected services that is feasible and meets the customer's requirements. On the lower layer, that is concerned with the individual schedules of the service providers, the services have to be scheduled into these individual schedules. Obviously, those two scheduling layers cannot be considered independently, but have to be resolved hand in hand to result in a global, feasible, and customer-satisfying schedule.

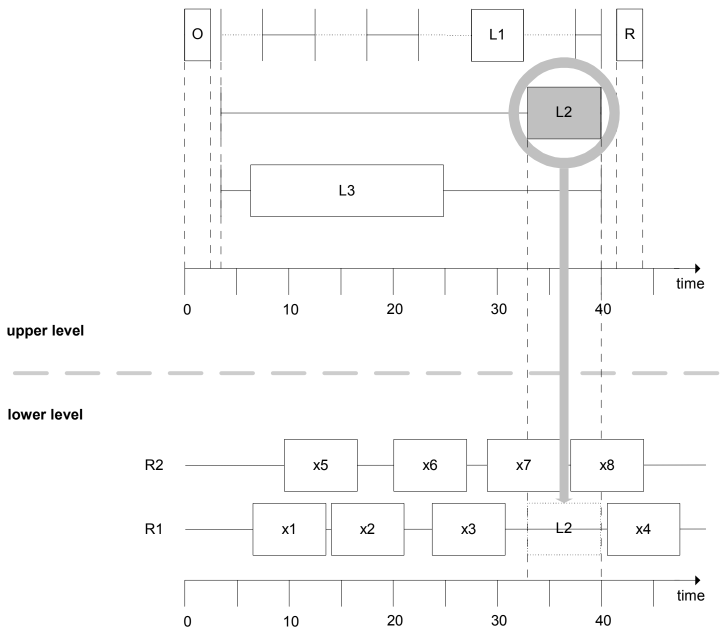

Figure 1 Two-layer scheduling problem

Take a look at the upper part of Figure 1. In this example of an upper level schedule, the customer has chosen an outward flight O and a return flight R, which mark the timeframe for his holiday stay. Between those two operations, the customer wishes to attend three different kinds of leisure activities L1, L2 and L3 in an arbitrary sequence. All three of them shall be carried out in the same timeframe, sometime between arrival and departure, and must not overlap. All three activities have a predefined duration. But while L2 and L3 can start any time, L1 can only be carried out within some predefined timeslots. In the figure above, the timeslots where activity L1 can be carried out are the solid lines between the vertical lines (plus the slot where L1 is temporarily scheduled), while the time spans where the service is not offered are represented by the dashed lines. The scheduler now has to find feasible sequences, i.e. a schedule for those operations within the given constraints. A feasible solution is already shown in Figure 1.

Now the scheduler can check with the service providers' schedules on whether there still is available space for the respective services at these times. As an example, the lower part of Figure 1 shows the lower level schedule of a service provider for activity L2. As can be seen, this service provider offers two resources R1 and R2 for this service, on which bookings are accepted. The boxes labeled x1 through x8 are other customers' reservations that were done previously. In this case, the desired booking of L2 is still possible on R1.

For every operation in the upper level schedule (i.e. O, R, L1, L2, L3), the scheduler has to check with the respective lower level schedule for booking possibility. Only then, the upper level schedule is valid in the way that besides fulfilling all customer wishes and restrictions, it could actually also be booked in that way with the service providers.

The idea now is to get all the relevant lower-level schedule information first, and integrate it into one composite scheduling problem, rather then first generating a feasible upper level schedule and then querying the lower level schedules on whether they still offer available space for that times. To do that, we suggest the following procedure that are based on the assumption of real-time interoperability, i.e. a permanent data connectivity between the virtual enterprise system and service providers, as done in other work like [2]:

```
1    receive to-be-scheduled operations O, R, L1, L2, L3
2    fetch needed parts of lower-level schedules
3    merge schedules
4    generate a feasible schedule
```

Figure 2 Sequence of a smart two-level scheduler

The integration of lower-level information into the upper-level scheduling problem yields the following, integrated scheduling problem:

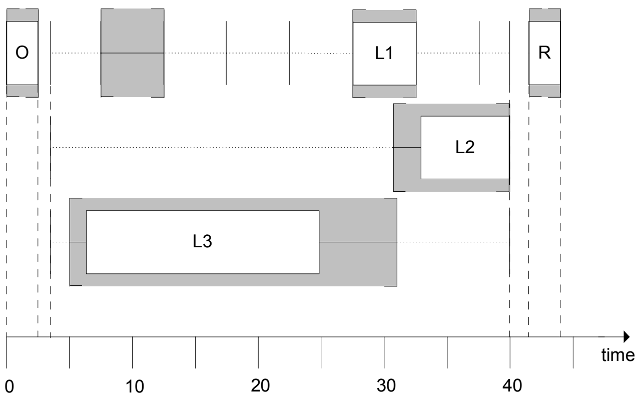

Figure 3 Composed single scheduling problem

The gray areas show the timeframes within which the respective operations can still be scheduled in the lower-level schedules. On the search for feasible solutions, the upper-level scheduler can move the respective operations back or forth within those gray areas. Until now we have assumed that service providers are indifferent about when to accept a booking, as long as scheduling takes place within the gray areas, i.e. the still available timeslots.

Proceeding one step further from here, we can now incorporate service providers' preferences. The desired result is to replace the plain gray areas with preference functions that assigns a preference value to each point of time of the available timeslots, representing the service providers' desire to receive a booking for that point of time.

3.2 Preferences

We assume the service provider's desired goal to be a full utilization of his resources. Therefore we link the preference function to the booking status, i.e. the number of still available resources at a given point in time. So the first step is to generate a function that represents this availability over the lapse of time.

With respect to this, we can distinguish two kinds of services. Those where individual bookings are done for points in time and do not overlap in time; and those where bookings can be done anytime, maybe even for an arbitrary duration, and hence potentially overlap in time on the different resources.

An example for the first case would be theater seats. It is easy to generate the availability function for this case. For each of the bookable points in time, the number of still available resources is added up, thus generating a discrete availability function over the lapse of time, as depicted in Figure 4:

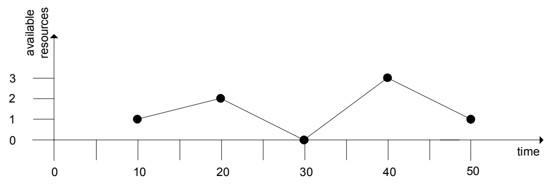

Figure 4 Discrete availability function

Let us say we are now at time zero in the figure above. A customer wishes to consume this service provider's service. He is indifferent about when to consume it, as long as it is sometime between time five and sixty. The exemplary service provider could prefer to schedule this customer in a slot at time forty over scheduling him in the last remaining slot at time fifty. The reason for that being, that if the customer was booked for the last spot at time fifty, and later another customer wishes to consume the service, but restricts his whish to only the slot at time fifty, the service provider would lose this second customer. If he had booked the first customer into a slot at time forty, he could have also served the second customer by booking him for the slot at time fifty, without penalizing the first customer.

To sum it up, the service provider could prefer booking customers into less booked times, over times where there already are plenty of reservations. Therefore, we could deduce a preference-function from Figure 4 that would basically look exactly like the availability function. Each possible timeslot t in the schedule for service S is given a preference value V. The preference-function then looks like this:

$$p_S(t) = V$$

In the example above, given

$$t = \{10, 20, 30, 40, 50\}$$

the functional values are

$$V = \{1, 2, 0, 3, 1\}$$

The second category of services are those where services on different resources overlap in time. Such services generate continuous availability functions. We will now focus on this tougher situation of overlapping bookings and present our algorithm for solving that problem.

Our solution is a dynamic approach. To find the available timeslots for every point in time, we have to take the already scheduled operations from their assigned resources and reassign them – that is, we potentially shift them to other resources, but don't move them back or forth on the time axis. By doing so, we want to maximize the available gaps between already scheduled services.

The rules for this algorithm are the following:

```
0a  put all operations into the repository
0b  consecutively number the resources, starting with number 1

 1  take the operation from the repository with the earliest start time
 2  put the operation on the resource with the lowest possible number
 3  renumber the resources:
3a      all resources with numbers up to the number of the resource the last operation has been put onto
3b      the resource with the operation with the latest finish time gets number one, etc.
 4  go to step #1
```

Figure 5 Rules of the dynamic approach

Let us go through that procedure by means of a new example and a couple of figures. Before the procedure starts, all operations are taken out of the schedule and put into a repository with no order (step #0b in Figure 5).

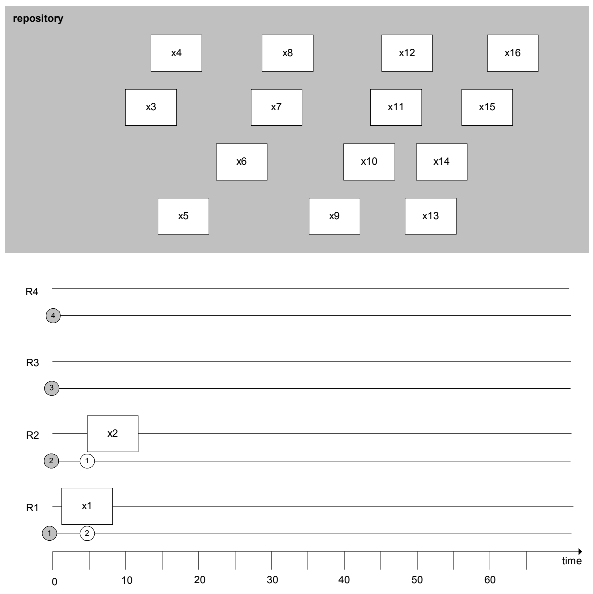

Figure 6 Reassigning the consecutive numbering (step #3 of the rules)

At the top of Figure 6 we can see the repository with the operations initially placed in it. We have taken them off the resources, but kept their position in time. The operations are numbered consecutively according to their starting times. That is, x1 has the earliest starting time, all the way through to x16 with the latest starting time.

We have chosen an example with operations of standardized durations. But the considerations are also fully valid for operations with variable durations.

According to step #0b, we consecutively number the resources. As can be seen in Figure 6, each resource, R1 through R4, has two horizontal lines. The upper line will be the line to place the operations on. On the lower line we will place balls with numbers in them, that show this consecutive ordering of the resources that is needed for the algorithm.

Following step #1 from the rules, we first take the operation with the earliest starting time from the repository – that is x1. Step #2 tells us to put that operation on the first available resource, starting with the lowest resource number. Since all resources are still completely free, we place operation x1 on resource R1.

Steps #3a and 3b only really have an effect if we had to place an operation on another resource but the one with the lowest number. So lets skip those steps for now. That brings us to step #4, which basically just tells us to loop back to step #1.

Figure 6 actually shows the situation of our example for the second loop. We have taken operation x2 from the repository, and had to put it on R2, because it overlapped with x1 on R1. So let's now examine step #3 of the rules. According to step #3a, we have to consider renumbering all resources with numbers lower or equal to the one we have put the last operation on. The last operation, x2, was put on R2, which has the number 2. So we will regard resources with numbers 1 and 2, which are resources R1 and R2.

Step #3b instructs us to find that one resource among the considered ones, that holds the operation with the latest finish time, and assign it the new number 1 – that is resource R2. Still in step #3b, we continue looking for the resource with the operation with the second latest finish time – that is resource R1, to which we assign the new number 2. Now we are done, and loop back to step #1. If we proceed following these rules, we complete our procedure with the result as seen in Figure 7. The upper part shows the final schedule that maximized the free time gaps. The lower part depicts the time gaps that are bigger than the to-be-scheduled operation.

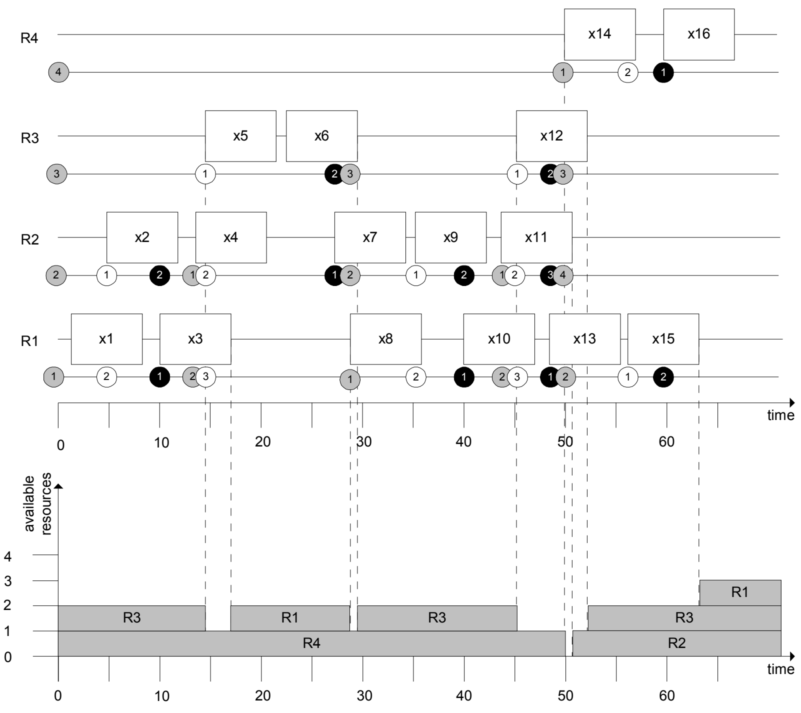

Figure 7 The schedule completion

3.3 The preference function

For a continuous preference function, we want to convert the availability function. We do that by evaluating every point in time of the availability function with respect to the quality of this position for being the starting time of the to-be-scheduled operation. This evaluation follows the single rule, that the quality of a position is determined by the minimum available resources within the time span of the operation. To illustrate this procedure, the following figures show the process of creating the preference function:

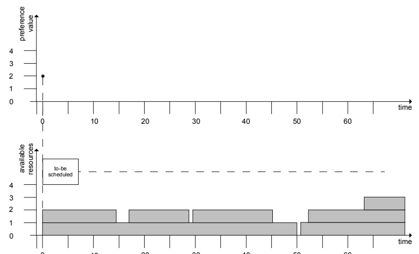

Figure 8 Step one

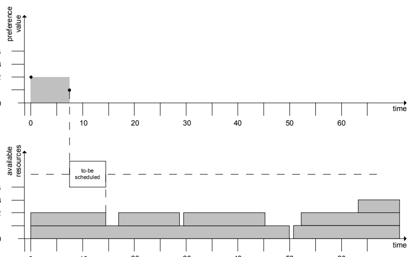

Figure 9 Step two

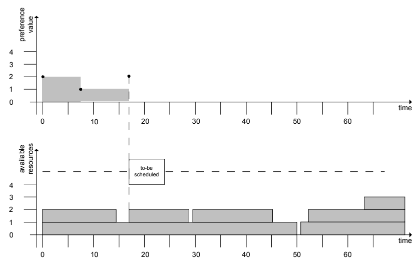

Figure 10 Step three

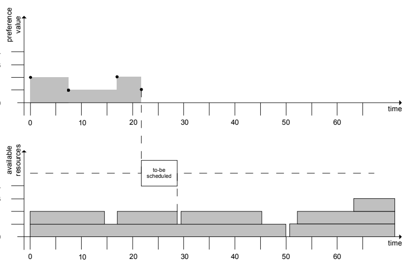

Figure 11 Step four

Starting with Figure 8, the to-be-scheduled operation is moved along the time axis. Following the mentioned rule, this procedure generates the preference function. Figures 8 through 11 show the to-be-scheduled operation at transition points of the availability function that affect the preference value. E.g. in Figure 9, there is a transition from two to one available resource at the end of the to-be-scheduled operation in its current position. According to the rule, this results in a preference value of one, starting at the time point at the beginning of the to-be-scheduled operation. In Figure 10, the to-be-scheduled operation again only spans parts of the available resource function with two available resources. Thus, the preference value skips back to a value of two.

The final preference function can then be seen in Figure 12.

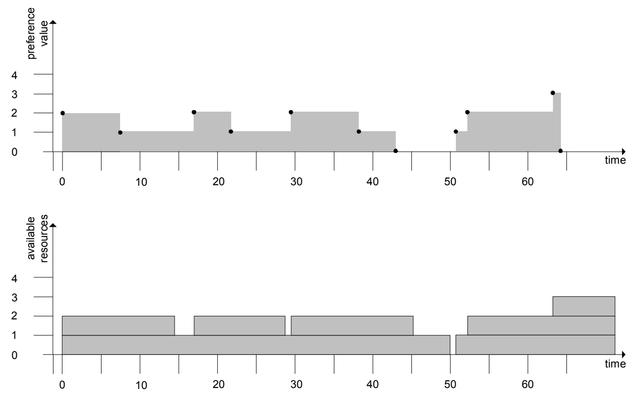

Figure 12 Final preference function

To sum up, for each point in time t, the function represents the quality of the position of the to-be-scheduled operation with starting time equal to t. The ranges of the preference function with value zero denote ranges where the service cannot be scheduled.

4 Conclusion and outlook

Feeding the preference functions into the upper-level scheduler, it can now try to find a schedule that maximizes the overall total of preference-values of all involved service providers. Let S be the involved services that are to be scheduled, n the number of involved services, t the scheduled time of a service, and p the preference-function of a service, then we can formalize the goal of the upper-level scheduler to be:

$$\max \sum_{i=1}^{n} p_{S_i}(t_i)$$

The result of all the concepts we have presented so far is one integrated scheduling problem that is solved to maximize the overall service providers' preferences. The customer defines his desired set of services with some restrictions with respect to ordering and times. The scheduler fetches the preference function for the considered timeframe from each service provider, upon which he then generates the optimal schedule.

Herewith, we have presented the basics for solving a scheduling problem in a virtual enterprise in the service sector with it's specific requirements to satisfy the service providers' interests with respect to their occupancy situation. An algorithm has been presented that returns a preference value based on a given schedule and a to-be-scheduled operation.

The next step will be to find the best algorithm, that uses the preference values to generate a schedule that maximizes the sum of the service providers' preference values. In future research, we will also elaborate on different kinds of service provider preferences, as well as concepts for easing up the tight restriction of not being allowed to move previously scheduled services back and forth in time.

Later, the concepts will be integrated in our touristic virtual enterprise system, that is being developed here in the EC3 MOVE (Management and Optimization of Business Processes in Virtual Enterprises) project as a scenario to try and prove our ideas and findings for the general field of virtual enterprises.

References

1. Larson, K., Sandholm, T.: Bargaining with Limited Computation: Deliberation Equilibrium. National Conference on Artificial Intelligence, Austin, TX (2000)
2. Camarinha-Matos, L. M., Afsarmanesh, H., Garita, C., Lima, C.: Towards an Architecture for Virtual Enterprises. 2nd World Congress on Intelligent Manufacturing Processes & Systems, Budapest (1997)
3. Kandler, F.: MOVE, Management and Optimization of Business Processes in Virtual Enterprises, Cooperation in a Virtual Enterprise in the Service Sector EC3 (2001); Requirements for a Virtual Enterprise in the Service Sector EC3 (2001); Conception of a Virtual Enterprise in the Service Sector EC3 (2001).