# Decomposition and Parallelization of Multi-Resource Timetabling Problems

Petr Šlechta

Czech Technical University in Prague, Department of Cybernetics,
Technická 2, 166 72 – Prague 6, Czech Republic
pslechta@ra.rockwell.com

**Abstract.** The timetabling problem consists in fixing a sequence of meetings between teachers and students in a prefixed period of time (typically a week), satisfying a set of constraints of various types. [1] Course timetabling is a multi-dimensional NP-Complete problem. [2] In this paper we present Multi-Resource Timetabling Problem (MRTP), which is our model for generalized high-school timetabling problem. The MRTP is a search problem – we try to find a feasible solution. Backtracking search was selected for solving MRTP. The main contribution of this paper is the Decomposition Algorithm for MRTP to parallelize the search, so the whole MRTP can be easily distributed and parallelized. Our approach was applied to a real life instances of high-school timetabling problems. Results are discussed at the end of this paper and parallelized search is compared with the centralized one.

## 1 Introduction

The traditional high-school timetabling problem is defined as follows [1] [3] (the terminology from the citations is used in this section; the chapter 2 defines the terminology which is used in the rest of this paper):

> We have $m$ classes $c_1$, ..., $c_m$, $n$ teachers $t_1$, ..., $t_n$, and $p$ periods 1, ..., $p$. We are also given a non-negative integer matrix $R_{m \times n}$, called requirements matrix, where $r_{ij}$ is the number of lectures given by teacher $t_j$ to class $c_i$. The problem consists in assigning lectures to periods in such a way that no teacher or class is involved in more than one lecture at a time.

This definition does not reflect the following requirements (see [1]):

(a) Some lectures require special rooms (e.g. musical education lecture requires a music room)

(b) Some lectures may be given to more than one class (e.g. gymnastic lesson may involve two classes together)

(c) Some constraints on timetables may be defined for teachers, classes, and rooms (e.g. some teacher may be unavailable at some time)

All requirements described above are generalized and considered in our Multi-Resource Timetabling Problem (MRTP) model, which is described in section 2.

We selected backtracking search as a technique for solving MRTP. To parallelize the search we have developed a Decomposition Algorithm (DA) which transforms

MRTP into non-oriented graph, applies color marking to the graph, and discovers how to decompose and parallelize the search. The algorithm is described in section 3.2.

Our approach was applied to real life instances. Example of search parallelization is given in section 3.5. In section 4 we discuss some results and we compare parallelized search with the centralized one.

## 2 The Multi Resource Timetabling Problem (MRTP) Model

In MRTP model we treat all teachers, classes, and rooms as ***resources***. Each resource has defined some constraints on its timetable.

All lessons are treated as ***events***. Each event requires some number of resources (e.g. a lesson of mathematics requires one teacher and one class, a gymnastic lesson may involve 4 resources: one teacher, two classes, and one special room).

Hence, the MRTP can be described by a set of resources and a set of events.

**Definition 1 (MRTP).** An instance of Multi-Resource Timetabling Problem is described by the tuple $INPUT = \langle E, R \rangle$, where $R = \{r_1, ..., r_m\}$ is a set of $m$ resources and $E = \{e_1, ..., e_n\}$ is a set of $n$ events to be scheduled.

**Definition 2 (Resource).** Each resource $r_i$ is described by the tuple $\langle S_i, C_i \rangle$, where $S_i$ is a timetable associated with the resource $r_i$, and $C_i$ are constraints for the resource $r_i$.

**Definition 3 (Event).** Each event $e_i$ is described by the tuple $\langle S_i, C_i, R_i \rangle$, where $S_i$ is a timetable associated with the event $e_i$, $C_i$ are constraints for the event $e_i$, and $R_i = \{r_1, ..., r_{m_i}\}$ is a set of $m_i$ resources required by the event $e_i$.

To schedule an event, all required resources must be properly scheduled to the same time slot with respect to all event constraints and all constraints of the required resources. Each event requires at least one resource (it may require more than one resource) for its completion. At one time, each resource can be occupied by one event at most. Event assignment is described more formally in the following definition.

**Definition 4 (Event assignment).** Event $e_i = \langle S_i, C_i, R_i \rangle$, where $S_i = [s_{i1}, ..., s_{iL}]$, $C_i = [c_{i1}, ..., c_{iL}]$, $R_i = \{r_{i1}, ..., r_{i,m_i}\}$, and $r_i = \langle S_j, C_j \rangle$, $L$ is length of the schedule, can be scheduled to the slot $x$ if all of the following conditions are satisfied:

(a) The slot $x$ of event's timetable is not occupied and is not forbidden by any event's constraint.

(b) All required resources have free slot $x$ in their timetables.

(c) None of the constraints of all required resources restricts us to use the slot $x$.

Our goal is to schedule properly all events $E$ from the input tuple $INPUT$. The following definition describes what we mean by feasible solution of MRTP.

**Definition 5 (Solution of MRTP).** A solution of Multi-Resource Timetabling Problem can be described by the set $OUTPUT = \{S_1, ..., S_m\}$, where $S_i$ is a timetable for resource $r_i$ from the tuple $INPUT$. All events from the $INPUT$ tuple must be scheduled according to the Definition 4.

## 2.1 Example of MRTP

In this section we use the MRTP model from the previous section to describe a simple high school timetabling problem. The problem is very simple and should only illustrate how the model can be used.

$$INPUT = \langle T, R \rangle,\ R = \{\ \text{A, B, C, John, Bill, Ray, Joe}\ \},$$

$$T = \{\ \text{M1, M2, F1, F2, H1, H2, H3, A1, A2, Ph1}\ \}$$

A, B, and C are classes (disjunctive groups of students), John, Bill, Ray, and Joe are teachers. M1, M2, F1, F2, H1, H2, H3, A1, A2, and Ph1 are subjects which should be scheduled. For simplicity we do not introduce any constraints on timetables in this example. Also the timetables are simplified: each timetable consists of 3 days, each day has 3 time slots.

Detailed description of all events is summarized in Table 1.

$$OUTPUT = \{\ \text{sA, sB, sC, sJohn, sBill, sRay, sJoe}\ \}$$

One possible solution of the problem described above is shown in Table 2. We present only timetables for the classes A, B, and C. Timetables for the teachers (sJohn, sBill, sRay, and sJoe) can be easily derived.

**Table 1.** Description of all events from the problem defined in this section

| *Event name* | *Event description* | *No of instances* | *Required resources* |
|---|---|---|---|
| M1 | Mathematics | 1 | A, John |
| M2 | Mathematics | 2 | B, Joe |
| F1 | Physics | 2 | A, John |
| F2 | Physics | 2 | B, Joe |
| H1 | History | 1 | A, Ray |
| H2 | History | 1 | B, Bill |
| H3 | History | 1 | C, Ray |
| A1 | Art | 1 | C, Ray |
| Ph1 | Philosophy | 1 | C, Bill |
| A2 | Art | 1 | C, Bill |

**Table 2.** One possible solution of the problem defined in this section – timetables for the classes A, B, and C. Timetables for the teachers can be easily derived

| ***sA*** | | |
|---|---|---|
| H1 | F1 | |
| F1 | | |
| M1 | H1 | |

| ***sB*** | | |
|---|---|---|
| M2 | | |
| M2 | | |
| H2 | F2 | |

| ***sC*** | | |
|---|---|---|
| A2 | | |
| A1 | Ph1 | |
| H3 | | |

## 3 Solution Approach

Backtracking search was selected as a technique for solving MRTP. To parallelize the search we have developed a Decomposition Algorithm (DA) which transforms MRTP into a non-oriented graph, applies color marking to the graph, and discovers how to decompose and parallelize the search.

### 3.1 MRTP as Non-oriented Graph

The MRTP is transformed to a non-oriented graph as follows:

(a) Each resource is represented as one graph node.

(b) Each event is transformed into a set of edges. The set of edges forms complete subgraph under all nodes representing resources required by the event to its completion.

Figure 1 shows a graph for one event which requires 4 resources for its completion (such event does not appear in our example). The example from section 2.1 transformed into non-oriented graph is shown in Figure 2.

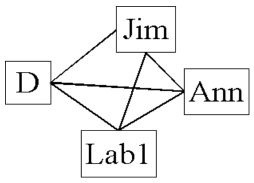


**Figure 1.** Graphical representation of one event which requires four resources for its completion (teachers Jim and Ann do teach together class D in room Lab1)

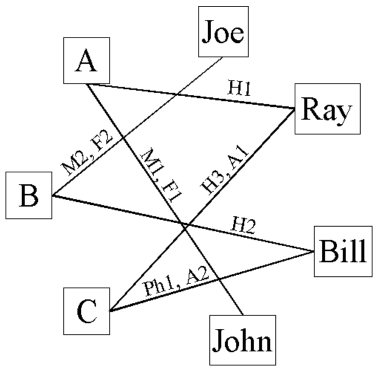


**Figure 2.** Graphical representation of the problem from section 2.1

### 3.2 Decomposition Algorithm (DA)

The parallelization of MRTP is based on splitting of the whole problem into independent subproblems. Splitting of the problem into independent subproblems corresponds to splitting the graph into isolated subgraphs (the subgraphs are not connected by any edge).

Usually we have to remove some edges from the graph to split it into isolated subgraphs. There are more sets of edges that can be removed from the graph to split it. Our goal is to select the proper set of edges which is small enough and which splits the graph into two subgraphs of approximately same size. We developed Color Marking Algorithm (CMA) which marks the graph with two colors and discovers the proper set of edges which should be removed from the graph. The Color Marking Algorithm is described in section 3.3.

Once we have found the proper set of edges to remove, we can start with decomposition of the problem. The removed edges correspond to events which must be scheduled to split the MRPT into two subproblems which are independent. These two subproblems may be solved in parallel and the found solutions may be easily integrated.

The graphical representation of example from section 2.1 is in Figure 2. If we remove the edge “H3,A1” the graph is split into two independent subgraphs – see Figure 3.

If we schedule events H3 and A1, the remaining events are split into two independent sets:

$$G_1 = \{\text{M2, F2, H2, Ph1, A2}\},\ G_2 = \{\text{M1, F1, H1}\}$$

All resources are also split into two sets:

$$G_{1r} = \{\text{B, C, Bill, Joe}\},\ G_{2r} = \{\text{A, John, Ray}\}$$

By scheduling an event from set $G_1$ only resources from set $G_{1r}$ may be affected. The same holds for sets $G_2$ and $G_{2r}$. The problem was split into two independent subproblems.

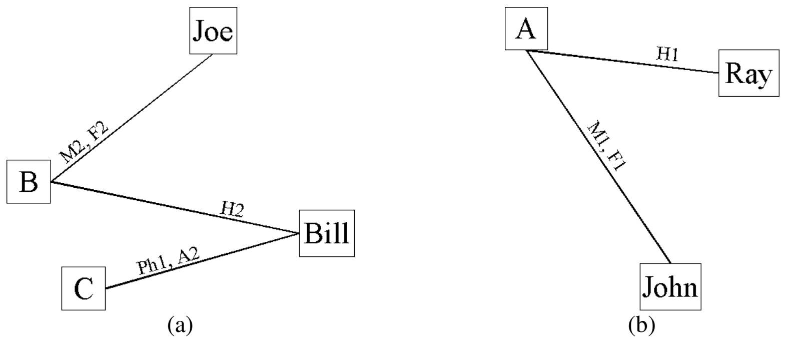


**Figure 3.** The graph from Figure 2 divided into two isolated subgraphs

## 3.3 Color Marking Algorithm (CMA)

This section describes the Color Marking Algorithm designed to discover the set of edges which should be removed to split the whole graph into isolated subgraphs.

CMA is based on the following idea: We remove one edge from the graph. One node, to which the removed edge was connected, will obtain red color and the second one will obtain blue color. Then the color is propagated through the edges and the color loses its intensity. When the color propagation is finished, each node has a portion of blue and red color. The node is finally colored by the color of which the node has bigger portion. At the end, red nodes form the first subgraph and blue nodes form the second one (see Figure 4).

If the graph has $z$ edges we start the CMA $z$-times to obtain all possible color markings of the graph. From all these color markings we select the most appropriate based on its rating. The rating composes of the two components: (a) number of edges which were removed from the graph (smaller number is preferred), and (b) difference between sizes of two subgraphs (the size of both subgraphs should be similar).When an edge is removed from the graph, the corresponding event has to be scheduled. This decision is made locally – a partial schedule is formed.

The Color Marking Algorithm is briefly described in the following listing:

```
for each node n in the graph {
    n.red_portion = 0;  n.blue_portion = 0;
}
for each edge e in the graph {
    remove e from the graph;
    node1 = the 1st node to which e was connected;
    node1.red_portion = 100;  node1.blue_portion = 0;
    node2 = the 2nd node to which e was connected;
    node2.red_portion = 0;  node2.blue_portion = 100;
    repeat {
        changed = false;
        for each node n in graph {
            call routine propagate_color(n);
        }
    } until (changed == false);
    count a rating for the colored graph;
}
select the partitioning with the best rating;

routine propagate_color(node n) {
    no_of_edges = number of all edges connected to the node n;
    red_portion_to_propagate = n.red_portion / no_of_edges;
    blue_portion_to_propagate = n.blue_portion / no_of_edges;
    for each edge e connected to the node n {
        neighbor_node = node to which the selected edge e is connected
        if (red_portion_to_propagate > neighbor_node.red_potion) {
            neighbor_node.red_portion = red_portion_to_propagate;
            changed = true;
        }
        if (blue_portion_to_propagate > neighbor_node.blue_potion) {
            neighbor_node.blue_portion = blue_portion_to_propagate;
            changed = true;
        }
    }
}
```

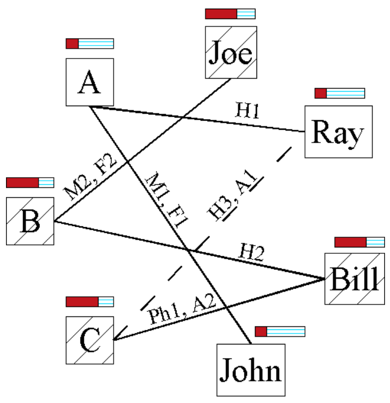
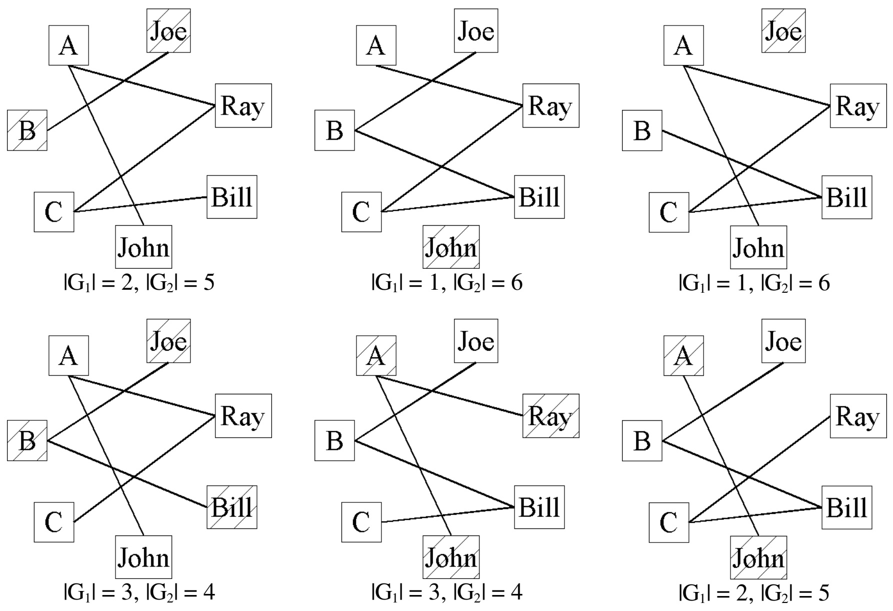


**Figure 4.** The colored graph from Figure 2 (assigned portion of red and blue color is above each node). The edge which must be removed to split the graph into two subgraphs is dashed. This edge corresponds to the evens H3 and A1. These events must be scheduled to split the whole problem into two independent subproblems

## 3.4 Examples of Graph Decomposition

All possible decompositions of the problem from section 2.1 are illustrated in the following figures.



**Figures 5.** Possible decompositions of the graph from Figure 2. Hatched nodes form the first subgraph, nodes without hatching form the second subgraph

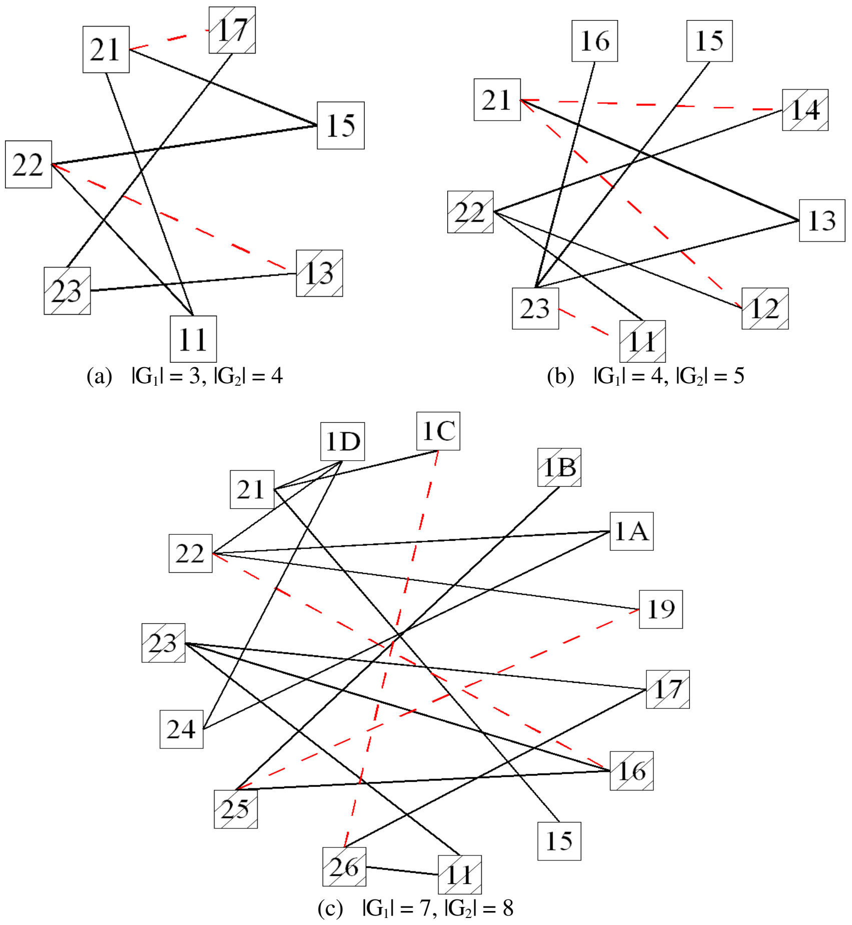


**Figure 6.** Some problems represented as graphs and decomposition of them. Dashed edges were removed to split the graph into two subgraphs. Hatched nodes form the first subgraph, nodes without hatching form the second subgraph

Decomposition of some other problems is shown in Figure 6. These problems are not described in this paper and the graphs should only illustrate decomposition of larger problems.

## 3.5 Example of Search Parallelization

The whole decomposition of the problem from section 2.1 is shown in Figure 7. At the beginning, we start with all events in one set. This set is received by the first scheduler, which schedules events H3 and A1 locally. After this the problem splits into two independent subproblems (events M2, F2, H2, Ph1, A2 represent the first subproblem, and events M1, F1 and H1 form the second subproblem). Then each subproblem is decomposed recursively as shown in Figure 7.

The parallelization of the search is shown in Figure 8. The whole problem was split into two subproblems, one was dedicated to the machine M2 and the second one was dedicated to the machine M4. The machine M2 (resp. M4) decomposed the given subproblem into two new subproblems, one was solved by machine M2 (resp. M4) itself, the second one was dedicated to the machine M3 (resp. M5).

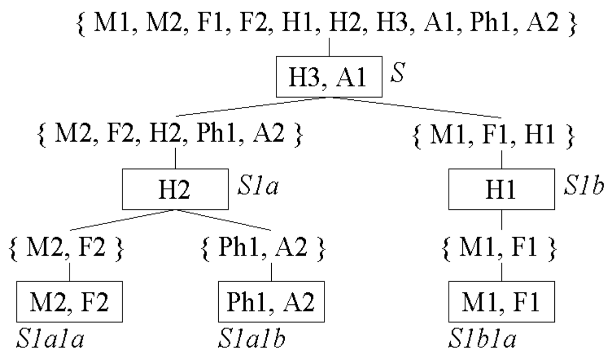


**Figure 7.** Decomposition of the problem from section 2.1. Boxes represent tasks which were scheduled locally by schedulers to split the given problem into subproblems

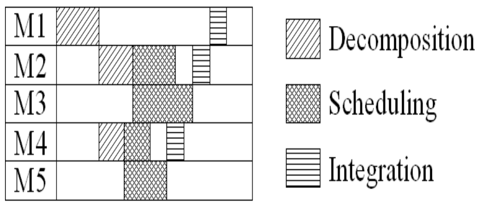


**Figure 8.** Parallelization of the search. Hatched boxes represent decomposition, scheduling, and integration of found solutions

## 4 Results

We have implemented a distributed scheduler written in Java 1.4 [7]. Some tests were performed to verify and exemplify the usability and performance of our approach. We run all these tests on PCs with Pentium III 866 MHz processor, 256 MB RAM, under Microsoft Windows 2000.

We would like to present results for some real life problems which were treated. Each problem description contains the complexity of the problem and the search times (for centralized and parallel search). Results are summarized in the following table.

**Table 3.** Some treated real life problems. Comparison of parallelized and centralized search

| ***Problem number*** | ***Complexity*** | | ***Search time*** | |
|---|---|---|---|---|
| | ***Resources*** | ***Events*** | ***Centralized*** | ***Parallel*** |
| 1 | 40 | 56 | 13.6 s | 11.2 s |
| 2 | 43 | 111 | 24.8 s | 21.4 s |
| 3 | 45 | 158 | 36.2 s | 28.6 s |

For example the problem number 3 is the problem of construction of timetable for high school with 9 classes, 36 teachers (some of them are external teachers), and 158 subjects. All the information needed for timetable construction was released by the school management.

Thanks to our decomposition technique, we were able to parallelize the whole search which resulted in the faster searching. We would like to optimize the Decomposition Algorithm in the future.

We would like to compare our algorithm to another algorithm based on decomposition, but we are not able to find any results we can compare. Interesting papers about decomposition algorithms are for example [4] [5], but they deal with another problems, which cannot be directly compared with our MRTP. We agree with paper [6] where the following conclusion is: "Although the STP (School Timetabling Problem) is a classical optimization problem, there is still no set of test-problems with the particular characteristics described in ... (at least, none that we know of to date) that can be used as benchmark."

## 5 Conclusions

We developed the model for a class of timetabling problems called Multi-Resource Timetabling Problems. We developed Decomposition Algorithm which allows us to parallelize the search. The approach was successfully evaluated, and it has the following advantages:

(a) The whole problem can be split into smaller independent subproblems which can be solved in parallel.

(b) Integration of found solutions is very simple and straightforward because the all subproblems are independent. There is no need for communication or synchronization between two subproblems during the search.

(c) The parallel search allows us to use modern grid computing techniques to solve our problem in a short time.

## References

1. Andrea Schaerf: A survey of automated timetabling, Report, CWI, 1995, ISSN 0169-118X
2. Michael W. Carter, Gilbert Laporte: Recent Developments in Practical Course Timetabling, in *Practice and Theory of Automated Timetabling II (PATAT'97)*, LNCS 1408, Springer, 1997, ISSN 0302-9743
3. Dominique de Werra: An Introduction to Timetabling, in *European Journal of Operational Research 19*, 1985
4. L. Friha, P. Queloz, C. Pellegrini: A Decomposition Method for Hospital Scheduling Problems, in *Workshop on Industrial Constraint-Directed Scheduling*, 1997
5. Hans-Joachim Goltz, Ulrich John: Methods for Solving Practical Problems of Job-Shop Scheduling Modelled in CLP(FD), in *Proceedings of Practical Application of Constraint Technology (PACT'96)*, 1996
6. Marcone Jamilson F. Souza, Nelson Maculan, Luiz Satoru Ochi: A GRASP-Tabu Search Algorithm to Solve a School Timetabling Problem, *4th Metaheuristics International Conference (MIC'2001)*, 2001

7. The Source for Java Technology, http://java.sun.com, Sun Microsystems, Inc., 2003
8. Vladimír Mařík, Olga Štěpánková, Jiří Lažanský, et al.: *Umělá inteligence (3)*, Academia Praha, 2001