An online approach to dynamic rescheduling for production planning.

Pierpaolo Caricato, Antonio Grieco

An online approach to dynamic rescheduling for production planning.

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2005-IJPR-0523.R3
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	17-Nov-2006
Complete List of Authors:	Caricato, Pierpaolo; Università di Lecce, Ingegneria dell'Innovazione Grieco, Antonio; Università di Lecce, Ingegneria dell'Innovazione
Keywords:	HEURISTICS, SCHEDULING
Keywords (user):	Constraint Programming

An online approach to dynamic rescheduling for production planning applications

Pierpaolo Caricato, Antonio Grieco

Dipartimento di Ingegneria dell'Innovazione

Università degli Studi di Lecce

Via Monteroni, Corpo O, 73100 Lecce, Italy

[pierpaolo.caricato,antonio.grieco]@unile.it

Tel. +39 0832 297806

Abstract

Manufacturing firms must often consider how to plan the production of a new order, containing the impact on the existing plan.

Two approaches are typically used to solve the problem: online scheduling and rescheduling. State of the art for both strategies is analyzed and methods available in the literature are proved to be inefficient for real world cases coming from production planning problems in a manufacturing firm.

We propose an alternative approach combining the most effective aspects of both traditional approaches. The already available production plan and the characteristics of the new order to be planned are studied and used in order to generate a new production plan that meets two requirements: containing the number of changes to the existing plan and minimizing the delays due to the newly planned order.

Constraint programming is used to implement the proposed approach. Results on case studies are also provided to evaluate the effectiveness of the proposed approach.

1 Introduction

Manufacturing firms in general and, especially, those pursuing a strict demand-driven strategy, need to use sophisticate Decision Support Systems (DSS) for production planning and scheduling. Effectively using the available resources in order to efficiently serve the market demands is, indeed, a key task for the firm management.

Even the best available commercial DSSs, however, cannot forecast the many unpredictable events that may lead the productive divisions to deviate from the scheduled plan.

Manufacturing operations, indeed, can be characterized by a wide range of uncertainties due, e.g., to the arrival of new orders, to the occurrence of machine failures, to uncertain raw materials arrival times, etc.. Production planning strategies are charged with accommodating such uncertainties in advance or reacting after the fact.

Over the last two decades a significant volume of research on the issues of scheduling with execution uncertainties has been developed, with application in several domains and using different approaches, see Aytug et al. [2005] and Herroelen and Leus [2005] for a detailed review.

The different existing approaches can be classified into three main categories: reactive scheduling, robust scheduling and predictive-reactive scheduling. The latter of these is by far the most studied and used.

In completely reactive scheduling, no attempt is made to understand and, hence, model the uncertainty that characterises the problem. Unpredictable events are just processed as they take place, using simple and rapidly applicable decision rules. A typical example of such approach is offered by techniques based on dispatching rules, for instance those presented in Bhaskaran and Pinedo, [1991], Haupt, [1989], Holthaus and Rajendran, [2000] and Ramasesh, [1990]. In these works, as well in many others available in the literature, each time a scheduling decision is needed, the available jobs are analysed and sorted according to some specific criteria, and the next job to be processed is selected according with the sorting order. These approach is typically effective when a

quick response is needed and overall optimisation objectives are of secondary importance or in the simplified cases in which these approaches can be proved to lead to an optimal scheduling.

As opposed to completely reactive scheduling, in robust scheduling no reaction to unpredicted events is provided, since the adopted scheduling tries to keep into account all the uncertainty. One of the most used techniques used to achieve such result consists in introducing several scenarios: a scenario is a possible and alternative outcome of several uncertain data. The most significant scenarios are designed: the scenario that is likely to produce the worst schedule is then identified as the worst case and the schedule evaluated under this scenario is adopted, since it is likely to perform better in most cases. Many works are available in the literature that follow this paradigm: e.g. Daniels and Kouvelis, [1995], Daniels and Carrillo, [1997], Kouvelis and Yu, [1997] and Kouvelis et al., [2000]. This approach requires a thorough analysis of the problem, in order both to identify meaningful scenarios and to chose the worst among them.

The third and most used approach consists in a trade-off between the other ones. In predictive-reactive scheduling, indeed, uncertainty is addressed both when the scheduling is generated and when it is executed. First, a predictive schedule is generated, which is optimal in terms of the selected performance index; then, this schedule is executed as long as unpredictable events do not take place. Once an unpredictable event has happened, a decision is taken whether to re-discuss the schedule or not.

A typical problem addressed through the predictive-reactive paradigm is the RTWSA (Real-Time Work Schedule Adjustment), defined as the modification of the planned work schedule on a real-time basis to cope with unexpected demand changes and/or disruptions of labor supply. The recent work Hur et al., [2004] on this topic represents the state-of-the-art for the solution of such problems. Actually, as also pointed out by the authors, not much work has been done on work schedule adjustment. Academic research has focused on how to develop “good” work tours/shifts and assign staff in advance of the day of service (see Brusco and Jacobs, [2000] and Easton and Rossin, [1991]). While managers in many industries (e.g. banking, restaurants and communications)

frequently conduct schedule adjustment decisions on a daily basis, existing labour scheduling literature has paid little attention to this issue. Recently, Hill et al., [2002] emphasized the need for research on the information needs, infra-structural processes, and economic consequences of real-time schedule control. Even if the work proposed by Hur et al., [2004] attempts to fill the void in labour scheduling research by exploring the nature of the adjustment decision and the corresponding factors that impact performance of the organization, its proposal for schedule adjustment is a rule-based approach that is capable of addressing the problem in the well-delimited field depicted in the work, but cannot be extended to include wider problems that still are related to schedule adjustment.

In this paper, we consider the need to build a new schedule in response to the arrival of a new job. We adopt a predictive-reactive approach but, focusing on the management of the arrivals of new orders. Hence, the "when to initiate a rescheduling action" issue is not to be considered. We rather focus our work on a way to execute a rescheduling action that best fits the characteristics of the considered problem.

In the literature, two main approaches can be found that address the specific task of including a new job within an already existing plan: online scheduling and rescheduling.

In the online scheduling approach, the current schedule is considered unchangeable and the incoming job is scheduled as early as possible after the already planned jobs. The methods that follow such an approach are extremely efficient in terms of computational effort needed to schedule the incoming job. These methods, indeed, are typically used in the field of scheduling jobs among processors in parallel computers. They can be considered as an extreme point in dynamic scheduling: online scheduling ensures the least (null) current schedule modifications, paid in terms of worst other performance (e.g. makespan, average/maximum lateness, etc.) deterioration.

On the other hand, rescheduling approach is the one used in the predictive-reactive paradigm. Hence, once that a rescheduling is decided, all the scheduled jobs in the current plan may be moved in the new schedule. Hence, this can be considered as the opposite extreme point in dynamic

scheduling: rescheduling ensures the best performances in terms of makespan or lateness minimization, but it leads the greatest amount of changes in the current schedule.

We propose an innovative approach that can be seen as an attempt to obtain a trade off between these extreme points: we try to obtain satisfying performances in terms of makespan or lateness reduction, while containing the number of changes in the current schedule.

The need for such a trade-off approach is extremely felt in the production planning field in manufacturing firms. In such a contest, indeed, once a production plan has been accepted, this starts up a complex series of activities strictly related with the accepted plan. As an example, a typical operations flow that is generated by the acceptance of a production plan may be as follows:

- print personalized reports of the production plan for each department involved in the production process;
- send picking orders to local or remote raw materials stores or suppliers;
- send order status feedback to customers.

Therefore, even though no planned order has yet begun to be physically processed, the production plan can be considered as preferably not modifiable once accepted. Hence, unless some urgent event happens, the accepted plan needs to be executed as is.

Nevertheless, it may happen that some unpredictable event takes place, that requires to change the accepted plan, in order, e.g., to satisfy an urgent new order.

This situation is very common in manufacturing firms that operate on a demand-driven base and this is where the approach we propose comes to help the decision makers, proposing a new plan that, though satisfying the urgent requirements of the incoming order, keeps the changes in the already accepted plan as low as possible. The price to pay in order to perturb as less as possible the accepted production plan, and hence the already started complementary activities, is the delay of some of the already planned orders with respect to the completion times they had in the previous plan.

The proposed approach has been validated on real world instances of the problem coming from a firm that produces shoes. The achieved trade-off has proven to be widely acceptable for the manufacturing firms to which the approach was proposed and whose real data provided the test cases used to validate the proposed algorithm.

The LIRS algorithm is described in section 2. The case study is presented in section 3, along with examples and computational results. Finally, conclusions and future issues are reported in section 4

2 Low-Impact Re-Scheduling

2.1 Problem statement

In manufacturing production management, the basic problem is to allocate machines and other resources such as tooling or operators, to jobs in order to optimize system performance, satisfying certain constraints such as due dates and capacity limits.

The term “schedule” is typically used in the literature to denote an assignment of machines to jobs for a specific period into the future, referred to as the “scheduling horizon”.

In manufacturing scheduling problems, a typical job is subjected to several processing stages. In this work we consider a generic hybrid flow shop, i.e. a system in which:

- N independent jobs are given
- available resources are organized in M stages
- each job is made up of up to M tasks corresponding with the available stages
- each stage can contain parallel identical resources
- each task has to be processed in the corresponding stage
- the tasks sequence is the same for all jobs, though not all tasks are present in every job

Each job j contained in the current schedule is characterized by the following data:

- the arrival time, i.e. the time the job enters the system
- the ready time , i.e. the time the job is ready to be processed on its first stage (for our purposes, we can consider this time always equal to the arrival time)
- the processing time t_{jk} needed the k-th task of the j-th job
- the start time scheduled for each task of the job
- the processing time required to process the entire job
- the due date, i.e. the time by which the last task required by the job must be completed
- the start time, i.e. the time when the first task of the job is scheduled to start
- the completion time, i.e. the time when the last task required by the job is scheduled to end
- the lateness l_j, i.e. the difference between the completion time and the due date; a positive value of this parameter denotes a delay

In this work, we consider the case in which a schedule has been generated for a given set of jobs. The schedule has been partially executed, when a new job arrives ("over time arrival") and has to be inserted in the existing schedule, under the hypothesis that no pre-emption is allowed. If the due date of the new job is early enough to make the new job "urgent", its insertion in the schedule will produce a perturbation. The objective is to minimize such a perturbation. Hence, the proposed approach can be labelled as "Low-Impact ReScheduling" (LIRS).

We define as "current schedule" the result of an already accomplished scheduling process that has led to a well defined plan for a given set of jobs. Such a plan represents the starting point for the LIRS algorithm. The "incoming job" is a new job to be added to the current plan so that the impact of its insertion is as low as possible. The "revised schedule" is the output of the LIRS algorithm and includes both the jobs already contained in the current schedule and the incoming job.

2.2 Feasibility, matching level and costs

The proposed approach is based on *swaps* between the current schedule and the incoming job. The incoming job will occupy, in the revised schedule, part of the resources that are used by a subset of the jobs in the current schedule. We call this a *swap*: we can indicate whether a swap is feasible or not and we can associate costs and benefits to any feasible swap. The subset of jobs to be substituted by the incoming job will be referred as the *replaced subset*.

Given a candidate replaced subset S for the swap with the incoming job i, the swap may be feasible if in every stage, the amount of time used by the considered replaced subset is greater or equal than the amount needed by the incoming job i, as indicated in (1).

$$\sum_{j \in S} t_{jk} \geq t_{ik} \tag{1}$$

This constraint is a necessary condition for the swap to be feasible under full resources saturation conditions. Indeed, in this case, if equation (1) is not satisfied, than the available resources freed by the replaced subset would not be sufficient for the incoming job to be processed. This is not necessarily true in general, i.e. it is not a necessary condition when not all resources are fully saturated in the current schedule. Anyway, this is a valid constraint for real world systems, in which the available resources are already reasonably saturated in the current schedule.

We define a swap that satisfies equation (1) as a *feasible swap*. A more accurate condition for a feasible swap to be actually performed is analyzed in section 2.4.

Given a feasible swap, a *matching gap* for the swap can be evaluated using the feasibility constraint: the replaced subset matches the new job as much better as much the inequality expressed in (1) tends to become equality, i.e. as much as the gap between the two sides of the inequality tends to zero. Hence, the matching gap μ_{Si} for the swap between the replaced subset S and the incoming job i can be defined as in (2). A perfect match would have a null matching gap: i.e. $\mu_{Si} = 0$.

$$\mu_{Si} = \sum_{k=1}^{M} \left(\sum_{j \in S} t_{jk} - t_{ik} \right) \tag{2}$$

The costs determined by a swap are caused by two circumstances:

- the replaced subset will generally leave unused resources when substituted by the incoming job
- both the jobs in the replaced subset and the incoming job will be generally characterized by a lateness

Both aspects are to be considered. Therefore, we can define two costs associated with the swap of the replaced subset S and the incoming job i.

The *unused resources cost* U_{Si} can be defined as in (3), where u_k is the resource cost per time unit for the k-th resource.

$$U_{Si} = \sum_{k=1}^{M} \left(\sum_{j \in S} t_{jk} - t_{ik} \right) \cdot u_k \tag{3}$$

The *lateness cost* L_{Si} for the same swap can be defined, as in (4), as the sum of all the lateness costs caused by the considered swap.

$$L_{Si} = \sum_{j \in S} \lambda_j l_j + \lambda_i l_i \tag{4}$$

The λ_j coefficient represents the lateness cost per time unit for the j-th job. Alternatively, this cost may also be defined as the average or as the maximum lateness caused by the considered swap.

2.3 Expected lateness estimation

Given the incoming job i and a replaced subset of jobs S, it is possible to estimate the impact of the swap in terms of lateness costs, without using the exact definitions given in (3) and (4). These approximations are necessary in order to improve the computational efficiency of the proposed algorithm.

As detailed in section 2.5, given the current schedule and the incoming job, we will initially solve the incoming job insertion problem using both the online scheduling and the rescheduling approach. We can then define the following parameters:

- the lateness l_i^{online} of the incoming job in the online scheduling revised schedule
- the lateness $l_i^{rescheduling}$ of the incoming job in the rescheduling revised schedule
- the lateness l_j^{online} of the replaced subset jobs in the online scheduling revised schedule
- the lateness $l_j^{rescheduling}$ of the replaced subset jobs in the online scheduling revised schedule

and, hence, using (4), the following lateness costs:

- the lateness cost L_{Si}^{online} due to the swap in the online scheduling revised schedule
- the lateness cost $L_{Si}^{rescheduling}$ due to the swap in the rescheduling revised schedule

As stated in section 1, $L_{Si}^{rescheduling}$ is the best lateness cost achievable, because it reconsiders the position of both scheduled and incoming jobs with the objective to optimize the system's performance. On the other hand, L_{Si}^{online} is the worst case for lateness costs, since it places the incoming job only after all other scheduled jobs have freed enough resources. Therefore, the following inequality holds:

$$L_{Si}^{rescheduling} \le L_{Si}^{LIRS} \le L_{Si}^{online} \tag{5}$$

In (5), L_{Si}^{LIRS} denotes the lateness cost due to the swap in the LIRS revised schedule. We can estimate this value as follows:

$$L_{Si}^{LIRS} = L_{Si}^{rescheduling} + \alpha\left(L_{Si}^{online} - L_{Si}^{rescheduling}\right) \tag{6}$$

where $\alpha \in [0,1]$ is a parameter of the algorithm to be set.

In a similar way, the lateness of the incoming job in the LIRS revised schedule can be estimated as follows:

$$\tilde{l}_i^{LIRS} = l_i^{rescheduling} + \beta\left(l_i^{online} - l_i^{rescheduling}\right) \tag{7}$$

where $\beta \in [0,1]$ is a parameter of the algorithm to be set.

The parameters α and β are, hence, responsible of the reliability of the lateness estimations made. A first testing phase on the problems used for the algorithm validation was conducted in order to find a method to set these values.

During this phase, 100 feasible swaps were generated for each test problem: for each swap, 11×11 (α, β) couples where generated, with each parameter varying in the [0,1] interval with a 0.1 step. The swap-inducted lateness estimations where made using the previous equations. The swap was, then, actually conducted and the estimations' errors were determined. As a result, the α and β values for each best swap estimations were collected and their average values where used throughout the following tests. The values found for the test were $\alpha = 0.64$ and $\beta = 0.28$.

This method can be used as a pre-processing step used to automatically fine-tune the LIRS algorithm on specific cases, before running it.

Finally, the impact of the swap in terms of lateness costs can then be estimated as:

$$\Im_{Si} = \left(\tilde{L}_{Si}^{LIRS} - L_S \right) + \tilde{l}_i^{LIRS} \tag{8}$$

i.e., the estimated impact of the swap between the replaced subset S and the incoming job i derives from two contributions:

- the increased lateness of the replaced subset;
- the estimated lateness $\tilde{l}_i^{LIRS}$ that the inserted incoming job is likely to have using the LIRS approach.

The former is given by the difference between $\tilde{L}_{Si}^{LIRS}$, the estimated lateness due to the Si swap using the LIRS approach, and L_S, the lateness the replaced subset S has in the current schedule.

2.4 Finite capacity resources

Once a feasible swap has been determined, the possibility to actually perform such a swap can only be determined considering the resources required both by the incoming job and by the replaced subset.

The resources that are to be thoroughly considered using the LIRS approach are the finite capacity resources. Infinite capacity resources that may exist are not involved in the considerations we make in this section.

Given a replaced subset S and an incoming job i, it has to be verified whether the resources freed through the elimination of the jobs in S allow the complete processing of the job i. In order to do so, each finite capacity resource has to be examined and its saturation has to be evaluated after the removal of the jobs in S. If the resource capacity is freed for a contiguous time interval in a manner that allows the processing of the incoming job on each machine, than the swap can actually be considered.

We will illustrate this concept with a simplified example in order to better explain its application.

Let us assume, for simplicity's sake, that each task of each job requires a single unit of each resource for all the duration of such a task.

Let us consider a replaced subset made of three jobs, A, B and C, and let us suppose that they all require a given resource that has capacity 2 for a time interval of two units.

A possible arrangement of the jobs in the current plan may be the one depicted in Figure 1, where a simplified Gantt chart and a saturation level for the resource over time are reported. The removal of the three jobs would lead to the saturation level over time reported in Figure 2.

Let us now consider the incoming job I, that requires the same resource for a duration of 6 time units. As it can be seen in Figure 2, the maximum number of contiguous time units during which the resource is not saturated is 5. Hence, the swap, though satisfying the constraint expressed by equation (1), cannot be actually performed.

2.5 The proposed algorithm

We propose an approach that exploits some aspects of both online scheduling and rescheduling approaches. We try to obtain a revised schedule that is as close as possible to the schedule that would be the result of an online scheduling approach, and hence as close as possible to the current schedule. Nevertheless, we try to keep the chosen performance indicator within a given range around the value that would be obtained through a pure rescheduling approach.

The revised schedule obtained through the pure rescheduling approach is used as the best case in terms of the performance parameter variation (makespan and average lateness). On the other hand, the revised schedule obtained through a pure online scheduling is considered as the worst case.

A commercial constraint programming solver (ILOG Solver (TM)) along with a scheduling class library that exploits the Solver features (ILOG Scheduler (TM)) is used to elaborate all schedules needed throughout the algorithm.

The input data for the algorithm are:

- a set of jobs fully characterized in terms of the parameters reported in section; 2.1;
- the current schedule for the considered jobs;
- an incoming job fully characterized in terms of the parameters reported in section 2.1.

The proposed algorithm is composed of the three steps described below. A block scheme of the proposed algorithm is reported in Figure 3.

STEP 1

We first evaluate the solution that we would obtain using a pure online scheduling approach, i.e. scheduling the incoming job as soon as possible after the already scheduled jobs.

STEP 2

We then determine the revised schedule that would be obtained through a pure rescheduling approach, i.e. re-elaborating all jobs, both already scheduled and incoming ones, to obtain a new schedule.

STEP 3

The core LIRS algorithm is executed. This consists of the three following sub-steps

STEP 3.1

The complete set of feasible neighbours of the current schedule is identified. The definition of feasibility given in section 2.2 is used here.

STEP 3.2

The neighbourhood determined at the previous step gets pruned. First, an estimation of the expected makespan and lateness are evaluated for each neighbour, as analyzed in section 2.3, allowing sorting the neighbourhood. Then, only a fixed number of best neighbours is considered, while the remaining ones is rejected.

STEP 3.3

The actual lateness is evaluated for each considered neighbour and the best one becomes the new schedule.

2.5.1 Step 1

A first, feasible solution can be found using a pure online scheduling approach.

Using this approach, all orders present in the current schedule are to be considered as unmoveable, i.e. the start and end time of each task of each job will remain the same in the revised schedule, while the incoming job will be scheduled as soon as possible using the resources capacities left available by the already scheduled jobs.

The solution found in this step will be the best feasible solution in terms of number of jobs that are modified in the revised schedule: no job from the current schedule is moved. On the other hand, this will be the worst case in terms of delay in the completion of the incoming job. Under this approach, indeed, the incoming job is only scheduled when the previous jobs leave a sufficient amount of unused resources to be used by the incoming job. This will obviously shift forward the completion of the incoming job, and this delay will be as much considerable as much the current schedule saturates the capacities of the resources.

2.5.2 Step 2

Another, feasible solution is found using a pure rescheduling approach.

Using this approach, none of the orders present in the current schedule are considered as unmoveable. Hence, in the revised schedule, the start and end time of any activity of any task might be arbitrarily different from the values it has in the current schedule, in order to better pursue the lateness minimization objective.

The solution found using this approach will hence be the best feasible solution in terms of lateness minimization, though it will be the worst case in terms of moved jobs.

The start and end time of each task of each job, both in the first solution, found in step 1, and in this solution, will be used throughout the algorithm as extreme limits needed for lateness estimations.

2.5.3 Step 3

The core LIRS algorithm consists of three subsequent steps that lead to the insertion of the incoming job into the current schedule pursuing the double objective of keeping as much as possible of the current schedule and delaying as low as possible the incoming job completion.

In order to achieve this result, the algorithm determines a neighbourhood of the current schedule, i.e. a set of feasible swaps that would allow the insertion of the incoming job into the current schedule.

Several approaches may be followed in order to generate such a neighbourhood. We propose a complete neighbourhood creation, i.e. we propose to create all the swaps that satisfy the constraint expressed by equation (1).

In sub-step 3.1 a constraint programming approach is used to generate all possible subsets of jobs that satisfy such constraint. The sub problem solved using the constraint programming solver consists in finding all the subsets of jobs present in the current schedule that have a combined duration that is greater or equal than the overall duration of the incoming job.

In sub-step 3.2, the neighbourhood gets pruned of the solutions that cannot be actually performed, according to the considerations made on finite capacity resources in section 2.4 and then it is further pruned using the lateness estimation methods described in section 2.3.

In the first part of this sub step, for each candidate swap, and hence for each candidate replaced subset, the resources saturation over time is calculated for each finite capacity resource after removing the jobs included in the replaced subset. We then verify if enough resource capacity has been freed for the incoming job to be processed. If this condition is verified for all the resources needed by the incoming job, then the swap is considered for the next phase of this step, else it is pruned.

All swaps to be considered in the second phase of this sub-step are both feasible in terms of satisfaction of constraint (1) and actually performable in terms of finite capacity resources requirements.

Finally, in sub-step 3.3, only the swaps that have passed the previous pruning phases begin to be actually performed, according to the estimated order, using the same scheduling software used to obtain the pure online and pure rescheduling solutions found in steps 1 and 2 of the algorithm. The evaluation stops when the processing time allowed by the user has been reached.

The problems solved by the scheduling software in sub-step 3.3 are very constrained problems, since all the tasks that do not belong to the incoming job or to the replaced subset are considered as unmoveable: i.e. their start/end time and their usage of the available resources cannot be changed. The scheduling software can only set the start and end time for the tasks that belong to the incoming job and to the replaced subset as well as their usage of the available resources, compatibly with all the constraints represented by the previous schedule of the other jobs. Hence, the processing time needed to perform the swap is much less than the time needed to obtain the first schedule. In the experimentation, it never took more than 1 second to perform the swap.

3 Case study

3.1 The industrial case

The proposed approach has been used to solve instances of production planning problem coming from a firm that produces shoes in the province of Lecce, in Italy.

The production of shoes is made up of several phases, which can be conducted on different production sites on a global scale. The main phases, present in all kind of shoes produced by the selected company, are detailed below:

- leather cut, in which leather hides are cut in order to obtain smaller shapes of leather, used in the following phases;
- sewing, in which leather parts get sewed together in order to obtain the vamp;
- assembly, in which the vamp, sole and other parts (strings, eyelets, labels, box, etc.) get assembled together in order to obtain the end-user product.

Besides these common, necessary phases, other optional phases exist, that are required only by subsets of the entire productions. Such phases are: handmade sewing for the vamp, handmade sewing for the sole, special machined sewing, pre-assembly preparation.

Furthermore, between some phases a quality control is required (e.g. on acceptance of sewed vamps coming from extern weavers) and expeditions are also to be planned when not all the production process is accomplished in a single facility (i.e. in more than 90% of the whole production).

We can consider each phase as an independent stage in a generic hybrid flow shop. Each phase, indeed, is realized in a parallel shop that usually involves several identical production lines. Furthermore, not all phases are required by all shoe types.

The firm adopts a mixed forecast and demand-driven strategy: i.e. a small group of orders are produced in advance, according to market trends in the previous seasons, while most items are produced in response of client orders.

The production is organized in two seasons: spring-summer and fall-winter. Clients orders are collected in separate stages: a first, relevant, amount of orders is collected during trade fairs and fashion events that take place before each season starts, while the rest of the orders arrive later, once the season has already begun.

In accordance with the orders collection, a first production plan is elaborated, before a season starts, that includes all early orders collected during trade fairs and already available before season production starts, along with a small group of forecasted orders. However, once the season production starts, more orders usually arrive during the production (over time arrivals).

The insertion of these new orders must fulfil the same constraints that are satisfied by the already scheduled orders. These orders refer to the current season and hence must be inserted in the current schedule. The objectives are to minimize the perturbation in the current schedule due to the incoming orders and to keep the system's performances, in terms of lateness or makespan, within a given range around the value they had in the existing schedule

3.2 An example

In order to better illustrate the logic beneath the proposed approach, a much simplified instance of the case study can be considered.

We suppose that the season's production is made up of 15 orders. All but one are available before the production is planned, while the last one arrives immediately after the production plan has been accepted. The due date of the incoming job is somewhere near the middle of the scheduling horizon.

We can also simplify the problem, considering only three, mandatory phases to be executed for each job (order).

This example is a much simplified version of the actually considered problem. However, it can be used to effectively illustrate the philosophy that drives the LIRS approach.

In particular, three important aspects of the problem are neglected in this example.

- All jobs require exactly the same three phases, while in the actual problem each job requires many more phases and not all jobs require the same ones.
- Only three resources are involved, one for each phase, while the actual problem involves more finite capacity resources.
- The arrival of the incoming job happens immediately after the production plan has been accepted. This means that no production activity has actually begun when the incoming job arrives, though the activities related with the acceptance of the plan may have partially or completely been executed (see section 1). Hence, no activity is to be considered as strictly unmovable. In the actual problem, incoming jobs usually arrive later during the planning horizon. In this case, all jobs phases that have already begun (or even completed) their processing are considered as unmovable (their start times become constraints for the scheduling process). This allows the new schedule to be coherently connected with the current schedule.

The current schedule is obtained considering the 14 available orders and scheduling them with the objective to minimize the makespan.

The last, incoming job is then scheduled following the online scheduling approach: as soon as possible after the already scheduled jobs. In this case, indeed, as described in section 2.5.1, no order is allowed to be changed in none of its tasks. The result of such strategy is depicted in the Gantt chart in Figure 4. Each job is composed by three, consecutive tasks, and is represented in the figure by three, interconnected bars. The horizontal axe, as usual, represents time.

In Figure 4, and in the following Figure 5 and Figure 6, the pale grey jobs are the current schedule. The dark grey job is the incoming one as scheduled by an online scheduling algorithm. As you can see in Figure 4, the incoming job is the last one to be completed: the system's performance is, hence, the worst possible. The only difference between the current schedule and the revised schedule, however, is represented by the incoming job.

On the other hand, Figure 5 shows the revised schedule (dark grey) that can be obtained using a pure rescheduling approach. Using this approach, as described in section 2.5.2, none of the jobs is considered to be unmoveable. Hence, the entire set composed of the previous jobs and the incoming job is treated as a new scheduling problem. The incoming job (the topmost in the figure) is completed according to its due date, as well as most of the other jobs: the system's performance is, hence, excellent. The number of changes from the current schedule (pale grey), however, is the highest. In other words, the insertion of the incoming job provokes a great perturbation in the current schedule.

Halfway between the two opposite approaches, the LIRS algorithm, as described in section 2.5.3, provides the trade off solution depicted in Figure 6. The incoming job (the topmost in the figure) is completed somewhat later than its due date. A single job (the fourteenth in the figure), however, among the others, is delayed in order to introduce the incoming one, determining a slight deterioration in the system's performance. The achieved solution is, hence, a good trade off between the two opposite solutions previously described.

In order to prove the effectiveness of the approach not only to solve this, simplified, example, but also complex, real world production problems, a thorough experimental campaign has been conducted, as reported in the following section.

3.3 Experimental tests

The proposed approach was tested on real data provided by the partner company. In particular, the orders of two accomplished production seasons were considered. Part of these orders are the ones received by the company prior to the beginning of the season, while the remaining ones are the ones received while the production for the season was being realized.

First, a schedule with all the orders available prior to the beginning of the season was obtained. Then, each order arrived later was used, along with its actual arrival time during the season, to

generate a new problem. The LIRS approach was used to obtain a new schedule including both the already scheduled orders and the incoming job/order.

The model used to formalize each scheduling problem includes the following characteristics:

- all jobs have a given due date;
- all jobs are composed of a set of related tasks, one for each production phase required by the particular shoe model to be realized;
- not all jobs are composed of the same number and type of tasks;
- tasks within an order are related through a set of temporal constraints that ensure the respect of the work cycle for each job;
- the duration of each task is calculated according to the number of products to be realized and to the speed of each phase for the particular shoe model;
- each task must respect resource constraints representing the connections between each phase and the physical resources it uses for its execution;
- expedition phases are also modelled and are characterized by time windows during which they may take place and by the capacities of the used containers;
- a multiple objective is pursued: the average lateness and the makespan are to be minimized; these objectives are linearly combined in the objective function using weights set by the user. In the experimentation, no particular emphasis was given to one of the objectives: both makespan and average lateness were normalized and the same weight was given to both the objectives.

The implemented scheduling software was used to obtain the first schedule, i.e. to schedule all the orders available before the season's production start. Using the Scheduler library, all precedence and resource related constraints were modelled using specific facilities, while the combined objective function was specified in terms of combined goals that drive the constraint programming solver

during its search for a good solution. The solution found is not necessarily optimal, since it uses a heuristic solution method.

In order to solve the problem generated by the arrival of a new job (the incoming job) to be inserted into an already existent and partially executed schedule (the current schedule), the following considerations were made:

- the arrival time of the incoming job, i.e. the actual time when the job became available during the season, represents the *present time* for the incoming job insertion;
- all the activities already started (completed or not) before the above defined present time must be considered as bounded, since no pre-emption is accepted by the company: i.e. their start/end times and their usage of the available resources are defined by the current schedule and they are constraints for the creation of the new schedule.

Each new order was then inserted within the current schedule using the proposed approach and comparing its results with the application of both the online scheduling and the complete rescheduling approaches. The obtained results became the new current schedule for the next problem to be solved in the same season, i.e. the problem generated by the next arrival of a new order.

Finally, a benchmark of the results achieved with the three compared approaches was realized, as detailed in the next section.

3.4 Computational results

We considered the production data of the last two concluded seasons as a benchmark test for the LIRS approach.

The first considered season included 74 order: 50 were available before the production's start, while 24 became available during the season's production. The second season included 88 orders: 59 available at the beginning of the production and 29 arrived later.

Each order is characterized by:

- the arrival time during the season (time 0 if already available at the beginning of the season);
- the order composition detail, i.e. the number of pairs of shoes to be produced for each model and size;
- the due date.

Each considered problem is peculiar, due to the many factors that characterize it, such as:

- the period of the season, that can be roughly classified into early-season, mid-season and late-season;
- the size of the new order, that is typically larger in early and mid-season and smaller later;
- the lateness of the production when the new order arrives, that is likely to be negative during early-season, small during mid-season and larger later;
- the saturation of the available resources, that is likely to be higher as the season proceeds.

We solved each problem using three approaches: online scheduling, rescheduling and LIRS. Each test was executed in less than 30 minutes. For each solved problem, we considered the following performance parameters variations between each algorithm and the current schedule:

- makespan
- average lateness
- number of moved jobs

The detailed results of the experimental tests are reported in Figure 7. For each test, a line in the table is present, with the following information:

- Test id (Test)
- Which of the two seasons the test belongs to (Seas)

- Performances obtained inserting the new job using the LIRS approach (LIRS columns)
- Performances obtained inserting the new job using the pure rescheduling approach (Rescheduling columns)
- Performances obtained inserting the new job using the online scheduling approach (Online scheduling columns)

Performance data for the three benchmarked approaches are represented by the following information:

- The makespan of the new schedule, obtained after the insertion of the new job (Mks)
- The percentage variation in the makespan between the current and the new schedule (Mks %)
- The average lateness of the new schedule, obtained after the insertion of the new job (Lat)
- The percentage variation in the average lateness between the current and the new schedule (Lat %)
- The number of jobs in the new schedule that have been moved in the new schedule, i.e. the number of jobs that are scheduled differently in the current and in the new schedule

Both makespan and lateness are expressed in the reference time unit used throughout the test, that was 4 hours (half a shift). The last line reports average values.

The computational results confirm the effectiveness of the proposed approach. In particular, we observe an intermediate deterioration of both the makespan and lateness performance indicators if compared with the ones achieved with the other approaches. Furthermore, the perturbation in the current schedule is extremely lower than the one produced by the pure rescheduling approach.

We focused on a predictive-reactive scheduling approach to the problem. Hence, no mathematical characterisation of the uncertainty has been considered. Using this approach, the two diametrically opposed online scheduling and rescheduling paradigms have been chosen to provide a complete and

valid benchmark for the proposed approach. Other completely different approaches may be followed to address the problem through a stochastic or fuzzy characterisation of the uncertainty, but they could not be compared with the proposed approach since they would need a deeper knowledge of the uncertain events than the one available in the considered problem.

3.5 Future research and developments

There are great research opportunities in the area that tries to fill the gap between theoretical scheduling models and real world of industrial production.

This work is just a first step towards this direction, addressing the particular problem that arises when a new, possibly urgent, order has to be inserted within an existing schedule.

In the proposed approach, several simplifying hypothesis have been made, though preserving enough complex aspects of the real problem to provide solutions that could be satisfactory and actually used in the production planning of the company. The removal of such hypothesis would lead to interesting developments of the present work and will probably be one of the research topics for our next works.

A first, major hypothesis that has been done is to have single arrivals of new orders: i.e. one single new order has to be inserted in the current schedule. Though not compromising the generality of the proposed solution approach, this hypothesis strongly limits the range of applicable solutions when multiple orders arrive at the same time.

The simultaneous arrival of multiple orders, indeed, can be modelled and addressed using the proposed approach through the expedient of simulating multiple arrivals of single new orders. This technique, though somehow leading to a solution to the problem, neglects aspects that could represent a good research line to further extend the proposed approach:

- ungrouping a batch of new orders into a sequence of single new orders leads to different solutions corresponding to the arbitrarily chosen sequence

- treating the new orders as a whole or as convenient sub-groups of new orders could drastically improve the proposed algorithm's approach performances

The second hypothesis that could be removed in order to improve the proposed approach is the deterministic knowledge of the durations of each modelled activity. As widely known, processing times are not always well predictable, especially when hand-made operations are required. A further development of the present work will deal with such uncertainty either in terms of stochastic or fuzzy modelled uncertainty.

4 Conclusions

We present an innovative approach for the solution of a problem that is often encountered in the manufacturing production planning field: the insertion of new orders in the production plan once this has already started.

The technique we introduce is an effective trade-off between the traditionally adopted online scheduling and rescheduling approaches.

The effectiveness of the proposed approach is proven by its application to real world instances of the problem.

The proposed approach leads high improvements in reducing the perturbations to the original schedule while determining negligible deteriorations in the system's performance indicators.

Future research on this topic will consider the introduction of uncertainty in the proposed approach, by means of either stochastic or fuzzy characterization of uncertainties or the possibility to achieve better performances by grouping incoming new jobs before inserting them in the current schedule.

5 Acknowledgments

This research has been funded by Italian Minister of Education and Research (MIUR) within the project FIRB titled "Architetture e tecnologie informatiche per lo sviluppo ed evoluzione di software open-source per la simulazione a componenti distribuiti, orientate al settore manifatturiero" (2001-2005 prot. RBNE013SWE PNR 2001-2003, FIRB art. 8, D.M. 199 Ric. 2001) and supported by AITeM (Associazione Italiana di Tecnologia Meccanica).

References

[Aytug et al. 2005]

Aytug, H., M. A. Lawley, K. McKay, S. Mohan, and R. Uzsoy (2005). Executing production schedules in the face of uncertainties: A review and some future directions. *European Journal of Operational Research* (161), 86-110.

[Bhaskaran and Pinedo 1991]

Bhaskaran, K. and M. Pinedo (1991). *Handbook of Industrial Engineering*, Chapter 83. John Wiley.

[Brusco and Jacobs 2000]

Brusco, M. J. and L. W. Jacobs (2000). Optimal models for meal-break and start-time flexibility in continuous tour scheduling. *Management Science 46* (12), 1630-41.

[Daniels and Carrillo 1997]

Daniels, R. L. and J. E. Carrillo (1997). B-robust scheduling for single machine systems with uncertain processing times. *IIE Transactions on Scheduling and Logistics* (29), 977-85.

[Daniels and Kouvelis 1995]

Daniels, R. L. and P. Kouvelis (1995). Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science* (41), 363-76.

[Easton and Rossin 1991]

Easton, F. F. and D. F. Rossin (1991). Sufficient working subsets for the tour scheduling problem. *Management Science 37* (11), 1441-51.

[Haupt 1989]

Haupt, R. (1989). A survey of priority rule-based scheduling. *OR Spektrum* (11), 3-16.

[Herroelen and Leus 2005]

Herroelen, W. and , R. Leus (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research* (165), 289-306.

[Hill et al. 2002]

Hill, A. V., D. A. Collier, C. M. Froehle, J. C. Goodale, R. D. Metters, and R. Verma (2002). Research opportunities in service process design. *Journal of Operations Management 20*, 189-202.

[Holthaus and Rajendran 2000]

Holthaus, O. and C. Rajendran (2000). Efficient jobshop dispatching rules: Further developments. *Production Planning and Control* (11), 171-8.

[Hur et al. 2004]

Hur, D., A. Mabert, and K. Bretthauer (2004). Real-time schedule adjustment decisions: a case study. *Omega 32*, 333-44.

[Kouvelis et al. 2000]

Kouvelis, P., R. L. Daniels, and G. Vairaktarakis (2000). Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions on Scheduling and Logistics* (32), 421-32.

[Kouvelis and Yu 1997]

Kouvelis, P. and G. Yu (1997). *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers.

[Ramasesh 1990]

Ramasesh, R. (1990). Dynamic jobshop scheduling: A survey of simulation research. *Omega* (18), 43-57.

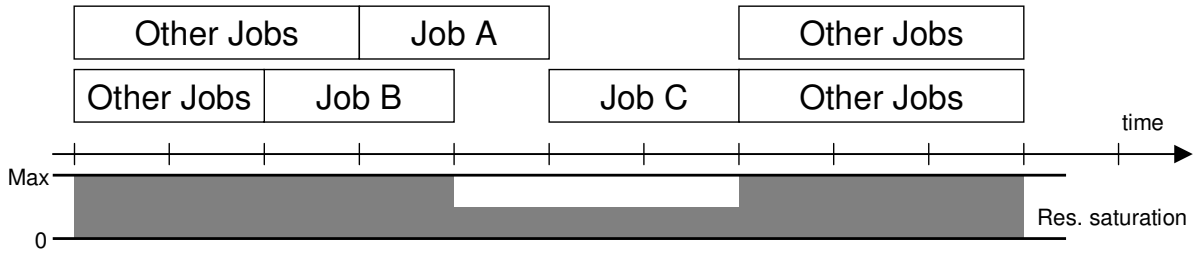

Figure 1: Gantt and saturation before removal of jobs A, B, C

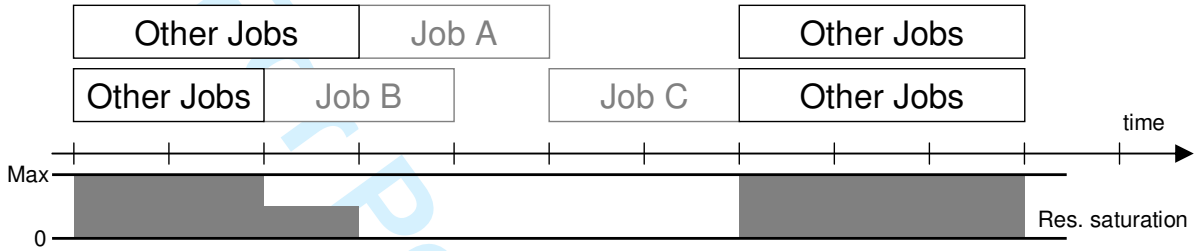

Figure 2: Gantt and saturation after removal of jobs A, B, C

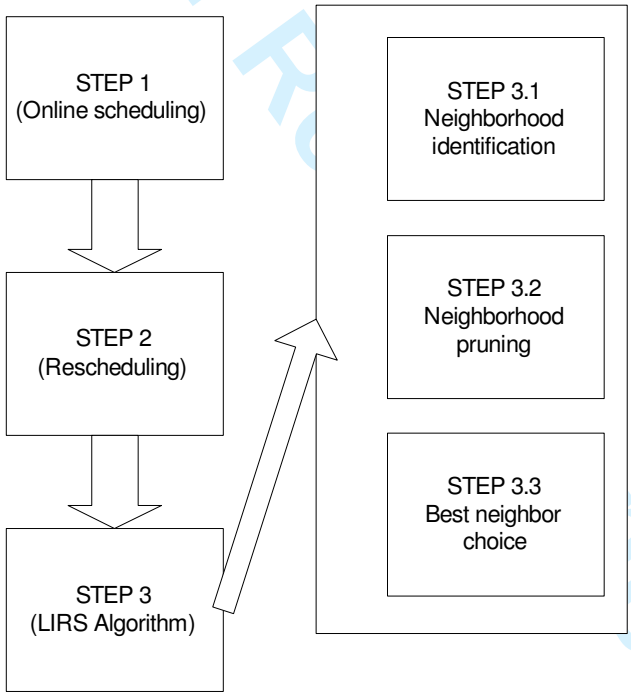

Figure 3: Proposed algorithm scheme

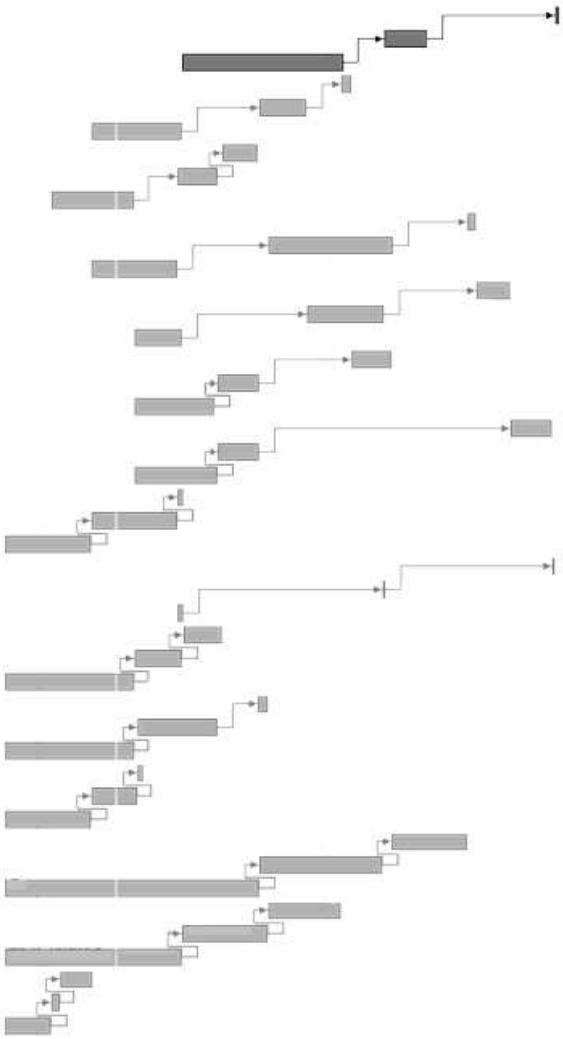

Figure 4: Current schedule vs. Online scheduling

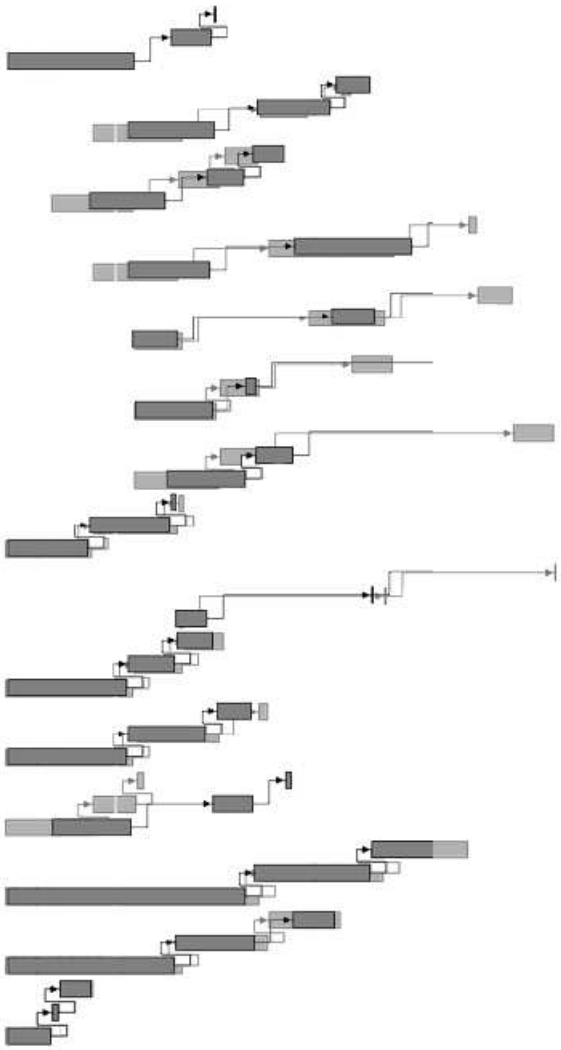

Figure 5: Current schedule vs. Rescheduling

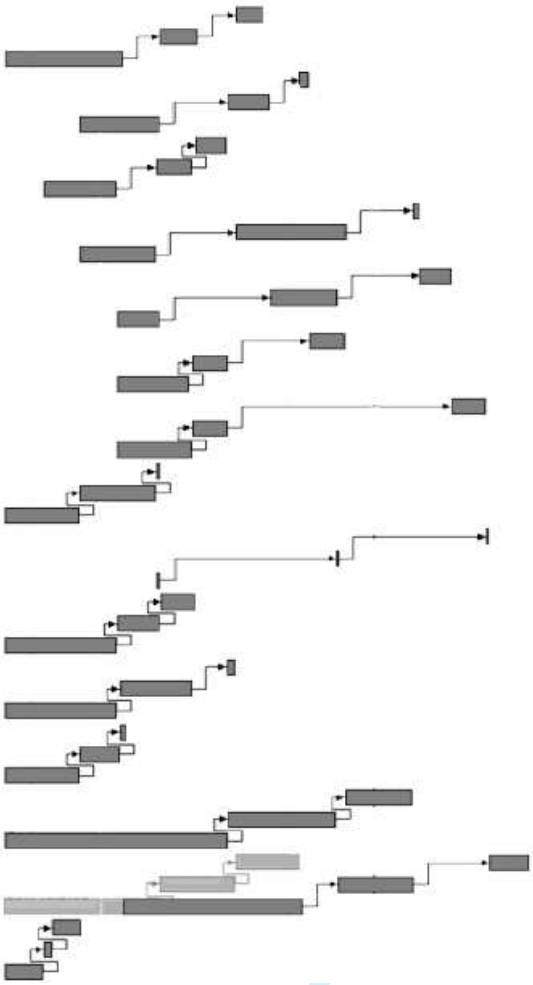

Figure 6: Current schedule vs. LIRS

Test	Seas	Current schedule		LIRS					Rescheduling					Online Scheduling				
		Mks	Lat	Mks	Mks %	Lat	Lat. %	Mov	Mks	Mks %	Lat	Lat %	Mov	Mks	Mks %	Lat	Lat %	Mov
1	1	238	5.37	240	0.82%	5.44	1.31%	2	238	1.98%	5.37	0.90%	17	242	1.80%	5.46	1.63%	0
2	1	240	5.44	245	2.26%	5.56	2.17%	2	240	0.21%	5.44	1.82%	8	244	1.77%	5.50	1.18%	0
3	1	245	5.56	251	2.12%	5.61	0.87%	2	245	0.79%	5.56	1.02%	11	248	0.90%	5.69	2.44%	0
4	1	251	5.61	256	2.03%	5.73	2.24%	3	251	1.70%	5.61	0.60%	15	256	2.28%	5.72	2.11%	0
5	1	256	5.73	258	0.84%	5.74	0.08%	3	256	0.02%	5.73	0.05%	5	256	0.09%	5.76	0.46%	0
6	1	258	5.74	262	1.56%	5.74	0.03%	3	258	0.57%	5.74	0.76%	10	261	1.41%	5.93	3.33%	0
7	1	262	5.74	265	1.21%	5.79	0.86%	1	262	1.11%	5.74	2.25%	12	263	0.63%	5.87	2.39%	0
8	1	265	5.79	268	1.09%	5.97	3.19%	2	265	1.14%	5.79	1.29%	12	270	1.71%	5.95	2.74%	0
9	1	268	5.97	270	0.79%	6.05	1.36%	3	268	1.08%	5.97	1.40%	12	270	0.86%	6.13	2.63%	0
10	1	270	6.05	271	0.24%	6.15	1.66%	2	270	1.51%	6.05	1.32%	14	275	1.91%	6.21	2.58%	0
11	1	271	6.15	272	0.56%	6.35	3.21%	2	271	1.51%	6.15	0.39%	14	271	0.10%	6.28	2.10%	0
12	1	272	6.35	278	2.22%	6.35	0.01%	1	272	0.79%	6.35	1.37%	11	281	3.07%	6.43	1.18%	0
13	1	278	6.35	284	2.08%	6.55	3.10%	1	278	0.78%	6.35	0.49%	11	283	1.73%	6.54	3.03%	0
14	1	284	6.55	285	0.35%	6.68	2.00%	2	284	0.90%	6.55	0.64%	11	292	2.87%	6.76	3.18%	0
15	1	285	6.68	288	0.91%	6.71	0.49%	2	285	2.00%	6.68	0.40%	18	285	0.09%	6.76	1.19%	0
16	1	288	6.71	290	0.88%	6.76	0.68%	1	288	0.34%	6.71	1.60%	9	290	0.87%	6.79	1.21%	0
17	1	290	6.76	296	1.94%	6.77	0.15%	2	290	1.84%	6.76	1.18%	16	297	2.21%	6.99	3.42%	0
18	1	296	6.77	300	1.41%	6.79	0.38%	2	296	0.55%	6.77	0.84%	10	302	1.97%	6.81	0.67%	0
19	1	300	6.79	305	1.63%	6.97	2.66%	2	300	1.10%	6.79	0.99%	12	304	1.46%	6.87	1.19%	0
20	1	305	6.97	309	1.35%	7.07	1.45%	2	305	1.68%	6.98	2.11%	15	313	2.81%	7.08	1.60%	0
21	1	309	7.07	313	1.15%	7.14	0.87%	2	309	1.07%	7.08	0.60%	12	318	2.94%	7.29	3.08%	0
22	1	313	7.14	316	0.99%	7.22	1.22%	2	313	1.43%	7.14	2.00%	13	321	2.59%	7.20	0.85%	0
23	1	316	7.22	321	1.82%	7.44	3.06%	3	316	0.11%	7.22	1.20%	7	321	1.70%	7.38	2.15%	0
24	1	321	7.44	322	0.21%	7.80	4.78%	2	321	1.52%	7.45	1.18%	14	327	1.61%	7.52	1.08%	0
25	2	211	4.32	214	1.18%	4.37	1.22%	1	213	0.97%	4.35	0.69%	8	214	1.50%	4.39	1.62%	0
26	2	214	4.37	216	1.14%	4.43	1.32%	2	215	0.70%	4.42	1.18%	11	217	1.59%	4.46	1.97%	0
27	2	216	4.43	219	1.21%	4.49	1.35%	1	217	0.50%	4.46	0.66%	9	219	1.43%	4.49	1.34%	0
28	2	219	4.49	220	0.67%	4.55	1.42%	1	221	0.99%	4.54	1.05%	9	221	1.12%	4.58	2.03%	0
29	2	220	4.55	224	1.86%	4.62	1.35%	2	222	1.04%	4.61	1.19%	11	223	1.31%	4.65	2.03%	0
30	2	224	4.62	228	1.74%	4.68	1.39%	3	227	1.09%	4.67	1.08%	8	228	1.87%	4.66	0.96%	0
31	2	228	4.68	229	0.64%	4.74	1.28%	2	230	0.97%	4.72	0.85%	12	231	1.32%	4.75	1.50%	0
32	2	229	4.74	232	1.19%	4.81	1.53%	1	232	1.09%	4.81	1.48%	14	233	1.47%	4.83	1.99%	0
33	2	232	4.81	235	1.25%	4.89	1.58%	2	235	1.02%	4.87	1.19%	9	235	1.38%	4.91	2.13%	0
34	2	235	4.89	238	1.15%	4.95	1.36%	2	237	1.00%	4.95	1.26%	12	238	1.41%	5.00	2.28%	0
35	2	238	4.95	241	1.17%	5.02	1.40%	3	242	1.77%	5.01	1.20%	10	242	1.63%	5.05	2.00%	0
36	2	241	5.02	243	1.12%	5.09	1.40%	2	243	1.03%	5.08	1.13%	14	245	1.76%	5.13	2.14%	0
37	2	243	5.09	247	1.55%	5.16	1.35%	2	246	0.98%	5.16	1.31%	10	247	1.50%	5.20	2.07%	0
38	2	247	5.16	248	0.40%	5.24	1.57%	1	250	1.05%	5.22	1.16%	13	250	1.21%	5.26	1.92%	0
39	2	248	5.24	252	1.47%	5.31	1.25%	1	250	1.00%	5.31	1.23%	8	253	1.85%	5.35	1.95%	0
40	2	252	5.31	254	1.09%	5.38	1.35%	2	254	1.08%	5.37	1.16%	14	255	1.52%	5.41	1.96%	0
41	2	254	5.38	257	1.17%	5.46	1.40%	1	257	1.06%	5.44	1.08%	9	258	1.58%	5.50	2.24%	0
42	2	257	5.46	260	1.18%	5.52	1.23%	3	260	1.11%	5.51	1.05%	8	262	1.62%	5.57	2.12%	0
43	2	260	5.52	264	1.21%	5.61	1.49%	3	263	1.11%	5.59	1.18%	11	265	1.78%	5.63	1.87%	0
44	2	264	5.61	267	1.18%	5.68	1.33%	1	266	0.95%	5.67	1.12%	10	268	1.50%	5.73	2.21%	0
45	2	267	5.68	268	0.49%	5.77	1.54%	1	269	1.00%	5.75	1.16%	11	270	1.24%	5.78	1.70%	0
46	2	268	5.77	273	1.88%	5.84	1.29%	1	271	1.03%	5.84	1.26%	14	272	1.64%	5.87	1.74%	0
47	2	273	5.84	276	1.27%	5.92	1.40%	2	276	0.97%	5.92	1.37%	14	277	1.40%	5.95	1.86%	0
48	2	276	5.92	282	1.99%	6.00	1.30%	1	280	1.13%	5.99	1.17%	12	282	1.99%	6.06	2.29%	0
49	2	282	6.00	283	0.40%	6.09	1.52%	3	285	1.13%	6.08	1.24%	10	285	1.06%	6.12	2.02%	0
50	2	283	6.09	286	1.17%	6.17	1.30%	2	286	1.05%	6.17	1.19%	12	287	1.44%	6.21	1.99%	0
51	2	286	6.17	290	1.30%	6.25	1.34%	4	289	0.99%	6.28	1.75%	11	291	1.64%	6.30	2.04%	0
52	2	290	6.25	295	1.66%	6.35	1.50%	2	291	0.29%	6.36	1.69%	12	295	1.66%	6.38	1.97%	0
53	2	295	6.35	297	0.68%	6.45	1.60%	1	296	0.34%	6.44	1.45%	12	299	1.36%	6.44	1.45%	0
AVG					**1.22%**		**1.48%**	**1.96**		**1.02%**		**1.15%**	**11.4**		**1.57%**		**1.94%**	**0**

Figure 7: Computational results