DYNAMIC RESOURCE CONSTRAINED MULTI-PROJECT SCHEDULING PROBLEM WITH WEIGHTED EARLINESS/TARDINESS COSTS[1]

M. Berke Pamay, Kerem Bülbül, Gündüz Ulusoy[2]
Manufacturing Systems Engineering Program
Sabancı University
Orhanlı, Tuzla 34956 Istanbul, Turkey

ABSTRACT

In this study, a conceptual framework is given for the dynamic multi-project scheduling problem with weighted earliness/tardiness costs (DRCMPSPWET) and a mathematical programming formulation of the problem is provided. In DRCMPSPWET, a project arrives on top of an existing project portfolio and a due date has to be quoted for the new project while minimizing the costs of schedule changes. The objective function consists of the weighted earliness tardiness costs of the activities of the existing projects in the current baseline schedule plus a term that increases linearly with the anticipated completion time of the new project. An iterated local search based approach is developed for large instances of this problem. In order to analyze the performance and behavior of the proposed method, a new multi-project data set is created by controlling the total number of activities, the due date tightness, the due date range, the number of resource types, and the completion time factor in an instance. A series of computational experiments are carried out to test the performance of the local search approach. Exact solutions are provided for the small instances. The results indicate that the local search heuristic performs well in terms of both solution quality and solution time.

1. INTRODUCTION AND MOTIVATION

Building a high rise building in a business district, or manufacturing a special purpose machine for a customer, or organizing a concert all involve various tasks to be completed in a systematic order to reach a final target. The project management approach can be applied to any of these endeavours as a decision tool to improve efficiency. This wide range of applications makes projects a common structure for organizing work. Besides internal company activities like maintenance or R&D, project based companies such as in construction, make-to-order manufacturing, or software development industries all present examples of multi-project management applications. Payne (1995) reports that up to 90% of the value of all projects occur in a multi-project context. Typically, multiple projects share common resource pools whose capacities are not sufficient to support all project activities at the same time, leading to the resource constrained multi-project scheduling problem (RCMPSP), which focuses on scheduling multiple projects while using available resource profiles and satisfying the precedence constraints to optimize the desired objective function.

Most project scheduling models are of static nature, where schedules are based on the

[1] To appear in "Essays in Production, Project Planning and Scheduling: A Festschrift in Honor of Salah Elmaghraby" P.S. Pulat, S. C. Sarin and R. Uzsoy, Springer Verlag, Berlin, 2013.
[2] Corresponding author: gunduz@sabanciuniv.edu

data that are available before the solution procedure and the effects of unexpected events such as disruptions in projects, arrival of new projects, and changes in resource availability are not considered. Herbots et. al. (2007) point out that static approaches are less realistic and a revision of the existing schedule might be required, especially when dealing with external projects. The main reason behind the dynamic nature of external projects lies in the complex network of business relations between companies. Cooperation with other organizations, subcontractors, and customers is a common way of doing business resulting in a multi-project environment. Anticipating the total project load in the future becomes almost impossible for the companies as their project portfolios change over time. Therefore, models dealing with the dynamic multi-project environments become critical to provide realistic decision instruments. The model presented in this paper is an attempt to partially fill the need for creating effective decision tools to be employed in dynamic multi-project environments; in particular, if the events have to be handled case by case with low visibility into the future.

Selecting the appropriate performance measure is essential to reflect reality. Minimizing the project completion time is a popular performance measure focusing on the effective usage of resources as well as the responsiveness of a company to its market. However, dynamic decision processes involve progressive schedule generation steps. Therefore, the starting times of the activities as well as the resource allocation decisions in the schedule can change dramatically while minimizing the makespan for the modified data sets. Handling these changes effectively requires organizational responsiveness – a crucial competitive capability. Drastic updates to the schedule and resource commitments may lead to significant organizational overhead and may not be desirable or even possible. Therefore, focusing on deviations from the baseline schedule in subsequent scheduling activities can help absorb any negative ripple effects of the dynamic events in the organization. As a result, punishing both earliness and tardiness, directly or indirectly, force the companies to schedule all activities on time or as close as possible to their due dates or completion times in the baseline schedule.

No baseline schedule exists for a newly arriving project, and the main concern for such a project is quoting a due date that trades off its potential revenue against the impact of accommodating it in the baseline schedule. Yang and Sum (1997) state that a negotiation procedure between the client (project owner) and the contractor is generally adopted in the decision process to handle this problem. The client wants the project to be completed as soon as possible and might even offer an increased payment for an earlier completion time as an incentive for the contractor. From the perspective of the contractor, the new project generates more revenue if completed earlier; however, the risk of paying late delivery costs for existing commitments has to be mitigated by pushing the new project toward the end of the existing schedule at the expense of forfeiting some of the potential revenue. The mathematical model we propose in this paper captures the trade-off between the revenue to be collected from a new project and the penalties which may result from not meeting existing delivery and resource commitments for the contractor.

The problem under consideration can be defined as follows. In a multi-project environment with a certain number of available renewable resource types; a processing time, a

due date, resource profiles, and associated unit tardiness and earliness costs are assigned to each activity. A baseline schedule exists for this set of projects. At a given point in time, a new project arrives. For the newly arriving project a due date has to be assigned and it has to be incorporated into the baseline schedule resulting in a new schedule. A cost parameter for the completion time of the new project representing the cost of delaying a new project by one time unit is defined and is referred to as the completion time factor K. The objective then becomes the weighted sum of the earliness/tardiness costs of the ongoing projects plus the cost associated with the new project's completion time. Hence, the problem under consideration can be considered as a variant of the resource constrained multi-project scheduling problem with weighted earliness / tardiness penalties (RCMPSPWET) and will be denoted as DRCMPSPWET in reference to the dynamic nature of the decision environment.

Within the context of this problem, the activity due dates and associated penalties are important parameters defining the characteristics of an instance. An applicable due date selection procedure is to convert the planned completion times into due dates. In other words, a baseline schedule, which is accepted by the contractor as well as by the client, is generated, and associated costs are defined to penalize deviations from the baseline plan in the new schedule. This approach can be applied to our deterministic model easily, since each disruption, as explained earlier, provides a new baseline schedule and can be converted into due dates for a potential new event in the future. With this approach, the dynamic problem can be simulated for multiple disruptions. The changes in revenue and deviations in schedules can be observed for multiple project arrivals at different points in time. Another strategy might involve defining some critical progress levels and penalties only for certain milestones of the projects. From a mathematical modeling point of view, defining milestones translates into choosing relatively higher cost parameters for the corresponding activities. Moreover, higher penalties for project completion times can be selected to emphasize the significance of completing projects at their previously scheduled times even if we allow shifting activities within a project. In the extreme case, we may omit the due date costs for all activities except those for the terminal activities of the projects. In summary, by setting the cost parameters associated with the activity due dates properly, we may model the problem with varying levels of flexibility and data requirements.

For any of these options, the following step is to determine the unit tardiness penalty values so that the deviations from the baseline schedule are not ruled out. An important factor for these penalties is the tightness of the due dates. A project with tight due dates has a greater possibility of becoming tardy; so the penalty values for a unit time should be lower than those under loose due dates, where the contractor has a wider time horizon to complete the project on time. In addition, the cost parameters have to be determined in a way that a trade-off between deviations from the baseline schedule and the due date of the new project exists.

In this paper, the dynamics of the problem are analyzed with respect to the total number of activities, the due date tightness, the due date range, the number of resource types, and the completion time factor. The goal is to design a solution method that rapidly provides near optimal solutions for this problem. Quick solution methods can make rescheduling time and cost feasible in comparison with repair heuristics, which incorporate myopic approaches in most

cases. This study makes the following contributions:

- The problem under consideration – DRCMPSPWET - is developed conceptually and a mathematical programming formulation of the problem is provided.
- A local search heuristic is designed and implemented. It is tested for solution quality and time against exact solutions obtained for a certain number of problem instances.
- A unique data set is generated for investigating the effects of the total number of activities, the due date tightness, the due date range, the number of resource types, and the completion time factor of the newly arriving project on the solution approach.

The paper is organized as follows: In Section 2, the related work in the literature and the problem definition are presented and an integer programming formulation for DRCMPSPWET is given. In Section 3, a heuristic approach for DRCMPSPWET is presented. The discussion of the data sets and an evaluation of the results are included in Section 4. Conclusions and possible extensions for future work are presented in Section 5.

2. RELATED LITERATURE AND PROBLEM DESCRIPTION

Herroelen and Leus (2005) classify the related work on DRCPSP under four categories: Reactive scheduling, stochastic rescheduling, fuzzy project scheduling and proactive scheduling. Note that our problem falls within the scope of the first category. Hence, we will concentrate here only on work in the area of reactive scheduling. Interested readers may refer to a recent review of stochastic project scheduling by Ashtiani et al. (2011). The models focusing on reactive scheduling try to model any unexpected event within a deterministic approach. Instead of executing a full rescheduling process, another option would be trying to minimize the effects of the unexpected event building on a baseline schedule which might or might not be repaired. One such example is the study by Artigues and Roubellat (2000) considering the case of activity insertion to the baseline schedule. The objective is to minimize the maximum lateness in a multi-mode multi-project setting. The multi-project environment is transformed to a resource flow network setting and dominant insertion cuts are used to generate the new schedule. El Sakkout and Wallace (2000) propose a method for minimizing the weighted absolute difference between the starting times of each activity in the baseline and modified schedules. The weighted absolute differences correspond to the earliness/tardiness concepts with symmetric costs, if the finishing times in the baseline schedule are treated as due dates. They propose a repair based heuristic approach to solve this problem.

2.1. Resource Constrained Project Scheduling Problem with Weighted Earliness/Tardiness Costs

To the best of our knowledge,the existing work on resource constrained project scheduling problem with weighted earliness/tardiness costs (RCPSPWET) is limited to single projects and no research has been conducted with multiple projects. Moreover, the concept of a baseline schedule is also not included in most of the studies. Neumann *et al.* (2003) mention an original schedule subject to change as a result of unexpected events. The limited work in the

literature includes some exact solution approaches as well as heuristic methods for the problem.

An exact solution procedure for the resource unconstrained version of the problem is suggested by Vanhoucke *et al.* (1999). The objective function is composed of the weighted sum of the earliness and tardiness values. This approach is based on a recursive search algorithm and consists of two main steps. First, a schedule is generated by scheduling activities at their due dates or later while considering only precedence relations. As a result, no right shift in the schedule can decrease the objective value. In the second step of the algorithm the set of activities, for which a backward shift can decrease the objective value, are selected by implementing a recursive search. Vanhoucke *et al.* (2001) extend the model to include resource capacity constraints. Using the exact solution algorithm for the resource unconstrained version they develop a branch and bound algorithm based on resolving the resource conflicts in a resource unconstrained solution. Precedence relations are added between activities in process during a period of resource conflict. Each conflict corresponds to a new node in the search tree and feasible solutions are obtained, if all conflicts are resolved. A further extension of the resource constrained model is provided by Vanhoucke (2002). In this study, for each activity, various due date options are offered. Each option differs in the tightness and unit cost values of the due date. That is, if an earlier due date is selected for an activity, the unit earliness and tardiness cost values are lower than those for a later due date. The objective is to select an appropriate due date option for each activity and generate a schedule such that the weighted sum of the earliness and tardiness values is minimized. A double branch and bound algorithm is developed to solve this problem. First, the resource unconstrained model is solved with the convex due date cost profiles. These profiles are obtained by converting the combination of different due date cost functions for each activity into a convex envelope using which a single due date is selected for each activity. However, unit earliness or tardiness costs might change according to the convex envelope profile. The solution yields a lower bound on the cost of the actual due date profile and the first branch and bound is applied while considering the distance between the convex envelope and the original due date profile for each activity completion time. The optimal solution is obtained after applying a second branch and bound procedure in order to resolve the resource conflicts as in Vanhoucke *et al.* (2001).

Ballestin et al. (2008) develop an iterated local search algorithm for RCPSPWET. A population of feasible solutions is generated and local search procedures are applied to improve the objective function value. Activity lists and a schedule generation scheme are used to generate corresponding schedules. The activities are scheduled iteratively with respect to a parameter called the simulated due date, which is the completion time of an activity in a randomly generated precedence feasible but resource unconstrained schedule. Simulated due dates are selected instead of the original due date values in the problem data in order to create diversity in the population. Four different local search procedures are then applied to existing schedules. At this stage, the activity lists are not changed; instead, schedules are modified in order to obtain improved solutions for a particular activity list in the population. To expand the search space, the activity lists are perturbed. The sequence of the activities in the list as well as the simulated due dates are updated using five different perturbation procedures.

Another list-based heuristic approach is proposed by Nanobe and Ibaraki (2006). This work covers a variety of project scheduling problems with convex cost functions including the weighted earliness/tardiness problem. The solution procedure relies on keeping event lists to obtain schedules. Each activity consists of a start- and an end-event, where positions of events in a list define priority relations. Each list can be mapped to an event-on-node network representation, and the dual problem can be solved as a minimum cost network flow problem. Event lists have to be resource and precedence feasible. This is done by checking the total resource demand of activities which are allowed to be processed simultaneously. If necessary, the list is modified and made feasible by changing the positions of events. A neighborhood is defined by moving events in the list backward or forward and an iterated local search is applied to the solution with the best objective value.

2.2. Problem Formulation

The DRCMPSPWET is defined here over an activity-on-node multi-project network with dummy start and finish activities. No precedence relation is assumed among the projects. The precedence relations among the activities are of type finish-to-start with zero time lag. All activities are of a single mode. Hence, only renewable resources are taken into account. Preemption is not allowed.

A special case of RCMPSPWET with a single project, a single resource of unit capacity, unit resource usage for each activity, no precedence relationships, and zero unit earliness costs reduces to the strongly NP-hard single-machine scheduling problem of minimizing the total weighted tardiness (Lenstra *et al*., 1977). Hence, RCMPSPWET is strongly NP-hard since the model presented in this study generalizes RCMPSPWET by incorporating a revenue function for the due date quoted for a new project. The overall objective is then to quote a due date that is as early as possible in order to maximize revenue while constructing a new schedule that minimizes the total weighted deviation of the activity finishing times from their completion times in the baseline schedule. We define the following notation.

Sets and indices:

T = set of time periods
I = set of all projects in the baseline schedule
I^* = set of all projects including the arriving project
h = $|I|$
$h+1$ = index of the arriving project
J_i = set of activities of project i
P_i = set of precedence relations between activities $j \in J_i$ of project i
R = set of renewable resources

Parameters:

W_{rt} = amount of renewable resource r available in period t

ES_{ij} = earliest start time of activity j of project i

LS_{ij} = latest start time of activity j of project i

d_{ij} = due date of activity j of project i

p_{ij} = processing time of activity j of project j

w_{ijr} = renewable resource requirement of activity j of project i of type r per unit time

e_{ij} = earliness penalty of activity j of project i per unit time

t_{ij} = lateness penalty of activity j of project i per unit time

K = completion time factor for the arriving project

The parameters presented above are required to define an instance of DRCMPSPWET. For each activity, the p_{ij} and w_{ijr} values define the single execution mode. However, there are additional parameters for activities depending on their status in the problem. For activities in the baseline schedule, a due date and unit earliness and tardiness penalties must be specified as well as a completion time factor standing for the cost associated with the completion time of the arriving project. Note that d_{ij} and K are not part of the original problem data in the experimental study. Their values depend on the baseline schedule of the instance. We elaborate on this issue further in Sections 4.1.4-4.1.6 and 4.1.8. Finally, the available capacities of the renewable resources are required. Note that the earliest and latest start times of activities can be calculated for a given time horizon $|T|$ using the conventional forward and backward pass algorithms of the critical path method (see, e.g., Badiru and Pulat (1995)). The objective function under consideration is nonregular, and delaying activities may decrease the total cost. Therefore, an optimal schedule may contain unforced idle time; however, no activity will complete at a time later than $|T|$ in an optimal schedule, where $|T|$ is set to the the sum of the maximum due date and the sum of the processing times of all activities of the arriving project.

Decision Variables

A 0-1 decision variable x_{ijt} is defined for each activity in the multi-project network including the dummy start and finish activities. For the activities in the baseline schedule, a finishing time, earliness and tardiness values have to be determined. For the arriving project, a due date is quoted as the finishing time of the dummy finish activity of the arriving project.

x_{ijt} = {1, if activity *j* of project *i* starts at time period t; 0, otherwise.

f_{ij} = finishing time of activity j of project i

d_{h+1} = due date of the arriving project

E_{ij} = earliness of activity j of project i

T_{ij} = tardiness of activity j of project i

Mathematical Model DRCMPSPWET :

$$\min \sum_{i \in I} \sum_{j \in J_i} \left(e_{ij} \cdot E_{ij} + t_{ij} \cdot T_{ij} \right) + K \cdot d_{h+1} \tag{1}$$

$$f_{il} - f_{ik} \geq p_{il} \qquad \forall i \in I^*, \forall (k,l) \in P_i \tag{2}$$

$$f_{ij} = \sum_{t=ES_{ij}}^{LS_{ij}} x_{ijt} \cdot t + p_{ij} \qquad \forall i \in I^*, \forall j \in J_i \tag{3}$$

$$E_{ij} \geq d_{ij} - f_{ij} \qquad \forall i \in I, \forall j \in J_i \tag{4}$$

$$T_{ij} \geq f_{ij} - d_{ij} \qquad \forall i \in I, \forall j \in J_i \tag{5}$$

$$d_{h+1} \geq f_{h+1j} \qquad \forall j \in J_{h+1} \tag{6}$$

$$\sum_{i \in I^*} \sum_{j \in J_i} \sum_{\theta = \max\{ES_{ij}, t - p_{ij} + 1\}}^{t} x_{ij\theta} \cdot w_{ijr} \leq W_{rt} \qquad \forall r \in R, \forall t \in T \tag{7}$$

$$\sum_{t=ES_{ij}}^{LS_{ij}} x_{ijt} = 1 \qquad \forall i \in I^*, \forall j \in J_i \tag{8}$$

$$x_{ijt} \in \{0,1\} \qquad \forall i \in I^*, \forall j \in J_i, \forall t \in ES_{ij}, \ldots, LS_{ij} \tag{9}$$

$$d_{h+1}, f_{h+1j} \geq 0 \qquad \forall j \in J_{h+1} \tag{10}$$

$$E_{ij}, T_{ij}, f_{ij} \geq 0 \qquad \forall i \in I, \forall j \in J_i \tag{11}$$

The objective function (1) consists of the weighted sum of the earliness and tardiness values of the activities in the baseline schedule and the completion time cost of the new project. Constraint (2) defines the precedence relationships among the activity pairs. The finishing times of the activities are determined in constraint (3). Constraints (4) and (5) determine the earliness and tardiness values, respectively. The quoted due date value, i.e., the completion time of the newly arriving project, is set by constraint (6). The total renewable resource usage in each time period is restricted to the maximum available amount in constraint (7). Finally, constraint (8) ensures that each activity is executed once and constraints (9), (10), and (11) define the domains of the decision variables.

This problem formulation above differs from the single project static RCPSWET problem formulation given by Vanhoucke *et al.* (2001) in that it reflects a multi-project dynamic decision

environment. The dynamic nature of the problem is incorporated into the formulation through the second term in the objective function (1) and the additional decision variables and associated constraints. Being the product of the completion time factor *K* and the quoted due date for the new project the second term represents an implicit cost of due date quotation and hence introduces into the formulation the trade-off between the stability of the activity finish times of the existing projects and the quoted due date for the new project.

3. AN ITERATED LOCAL SEARCH APPROACH FOR RCPSPWET

Heuristic procedures have been developed for RCPSPWET in single project environments as discussed in Section 2. List-based heuristics reported by Ballestin and Trautman (2008) and Nanobe and Ibaraki (2006) perform well both in terms of solution quality as well as computation times. Moreover, neighborhoods can easily be defined for the schedules represented by the lists and the associated schedule generation procedures are simple and efficient. Therefore, a population based local search procedure is suggested to solve the problem at hand. The general flow of the solution algorithm is presented in Figure 1.

```
input  : An instance of DRCMPSPWET.
output: A feasible solution for the instance.
begin
    Initialization;
    Create the initial population;
    foreach activity list in the initial population do
        Apply List Positional Neighborhood Search;
        Apply Timing-Based Neighborhood Search;
        /* The following lines are performed on some activity lists only.
           See text.                                                    */
        Construct an extended precedence graph that prevents resource infeasibilities
        based on the current best schedule associated with the activity list;
        Solve the optimal timing problem for this extended precedence graph as an LP;
    end
    Report the best schedule identified;
end
```

Figure 1: Flow of the local search heuristic.

The heuristic method starts by generating an initial population of activity lists. Three different improving steps are applied to this initial population iteratively in order to improve the activity lists. These steps replace the sequencing and optimal timing procedures commonly used in the machine scheduling literature for weighted earliness tardiness problems. (Kanet and Sridharan (2000) give an overview of different optimal timing algorithms in the machine scheduling domain.) First, a list-position based neighborhood search is performed to improve the sequencing in each activity list. An optimal timing based neighborhood search is then

applied to move chains of activities earlier in time. Finally, for all resource types in an instance, the associated arcs that prevent resource conflicts are added to the network and the resulting optimal timing problem is formulated and solved as a linear program (LP).

3.1. Activity Lists and Schedule Generation

An activity list in the population is used to represent a schedule. Each activity is assigned to a position in the list. In a precedence feasible activity list, each activity is positioned after its predecessors and before its successors. Given a precedence feasible activity list, a locally optimal schedule is generated by scheduling each activity in the list to start at its locally optimal position. For an activity in the baseline schedule, a locally optimal position is defined as the one which minimizes (earliness + tardiness) cost for this activity without shifting the activities already scheduled. The activities of the newly arriving project are scheduled as early as possible because the associated cost component in the objective function is increasing in the completion time of this project.

3.2. Initial Population Generation

An initial population is generated to apply the neighborhood search procedures. Each member of the population is a precedence feasible activity list. To ensure the diversity of the initial population and explore a larger portion of the search space, activity lists are constructed by applying two different priority rules and adapting a shifting bottleneck (SB) based heuristic originally developed for job shop scheduling problems with non-regular objectives by Bulbul and Kaminsky (2010) to our problem, in addition to randomly generating precedence feasible activity lists.

To create activity lists the most total successors (MTS) and minimum latest start time (LST) priority rules are employed by selecting the activity with the best value among the precedence feasible candidates. These are network and critical path based priority rules, respectively (Demeulemeester and Herroelen, 2002). The basic idea behind the selection of these dispatching rules is to increase the possibility of adding a larger number of precedence feasible activities to the candidate list earlier and thereby improving their chance of on time scheduling as well as achieving higher resource utilization. Biased sampling versions of these priority rules are also used to increase the size of the population. That is, candidate activities are assigned probabilities proportional to their respective priorities, and the next activity in the list is picked randomly based on these selection probabilities.

The SB heuristic is a well-known machine-based decomposition method in the machine scheduling literature (Adams et al., 1988). In the application of the SB framework to job shop scheduling problems, the machine capacity constraints are initially all relaxed, and are then added back to the problem sequentially by solving a series of single-machine scheduling subproblems. The objective function value of a single-machine subproblem provides an estimate of the effect of the capacity restrictions of the machine under consideration on the overall schedule. The currently unscheduled machine with the highest subproblem objective value is referred to as the bottleneck machine, and the sequence of operations on this machine

is fixed first before those of the remaining unscheduled machines. The SB approach was originally developed for the classical job shop scheduling problem of minimizing the makespan by Adams et al. (1988), and it was later extended to job shop scheduling problems with maximum lateness (Demirkol et al. 1997) and total weighted tardiness minimization objectives (Pinedo and Singer, 1999; Singer, 2001; Mason et al., 2002) among others. Recently, Bulbul and Kaminsky (2010) extended this framework to job shop scheduling problems with any objective function whose associated optimal timing problem can be expressed as an LP. Their approach is particularly effective, if the individual completion times are associated with explicit costs as in our problem. Based on this observation, we adapted the SB algorithm of Bulbul and Kaminsky (2010) for our purposes. Initially, a schedule is obtained by relaxing all resource capacities and solving the resulting model as an LP. The SB heuristic then resolves the resource conflicts present in the optimal solution of this relaxation iteratively by solving a set of single-resource weighted earliness tardiness scheduling subproblems with precedence constraints. The unit earliness and tardiness costs in the subproblems are estimated using LP sensitivity analysis as in the original paper. The subproblem is a generalization of the NP-hard single-machine weighted earliness tardiness problem, and the iterated local search approach we design for the overall problem is also used to solve the subproblems of the SB heuristic with some minor modifications and simplifications. These details are discussed in Pamay (2011). The solution of a subproblem introduces new precedence relationships based on the concept of resource flows (e.g., see Artigues and Roubellat, 2000). These new precedence constraints are incorporated into the optimal timing LP and ensure that the capacity of the resource under consideration is no longer violated. These steps are repeated until all resource conflicts are removed and a feasible solution to the original problem is obtained. This basic algorithm is enhanced by executing a restricted tree search over all possible orders of resolving the resource conflicts and results in several feasible solutions for the original problem. A standalone application of this SB heuristic does not produce high quality solutions; however, it provides us with a tool to diversify the initial population. The schedules constructed by the SB heuristic are converted to activity lists based on the activity start times and added to the initial population. In our computational study, we report the results of the iterated local search algorithm both with and without the initial solutions from the SB heuristic and demonstrate a significant added value from their inclusion in the initial population.

3.3. List Positional Neighborhood Search

Once the initial population has been generated, the first neighborhood search procedure starts. This process is applied to each member of the initial population separately and if an improvement is observed, the activity list is replaced and the search for better schedules continues with the new activity list. First, all activities in an activity list are sorted in non-increasing order of their contributions to the objective function. For the activities of the new project this contribution is zero unless they belong to the critical path of the project. A critical activity of the new project is assigned a cost of ($K\ f_{h+1,|Jh+1|}$), where $f_{h+1,|Jh+1|}$ is the completion time of the new project. The neighborhood search proceeds by processing each activity in the list in the order specified above. The activity under consideration may be moved to an earlier position in the list while preserving precedence feasibility. Consequently, it can be scheduled at

earlier stages of the schedule generation process and has a greater chance of incurring a lower cost. A selected activity may be moved anywhere between its predecessor with the latest position in the list and its current position. Each of these possible moves is evaluated by removing the activity from its current position and inserting it at the required spot in the list. For each position, the objective function value is determined by using the locally optimal scheduling scheme. If the objective value can be improved, this change is applied to the activity list. If the evaluated moves for the current activity are non-improving, the activity with the next highest cost contribution is selected and the procedure is repeated until a limited number of non-improving steps is reached. If no improvement can be observed until reaching this threshold level, the best non-improving move is applied and the move is added to a tabu list to track forbidden moves. In general, the neighborhood search for an activity list terminates, if either a prespecified maximum number of neighborhood search moves or a prespecified maximum number of non-improving steps is reached first.

3.4. Timing-Based Neighborhood Search

To check for further possible improvements a timing-based local search is applied. The locally optimal scheduler places an activity in its locally optimal position without shifting activities already scheduled. Therefore, the total objective function value may be reduced by moving a single activity earlier or later in time. This can be done by modifying the due dates of the activities temporarily such that the locally optimal positions of the activities are changed for the same sequence. To this end, we first determine chains of activities in the precedence graph which are processed without idle time in between in the current schedule and then calculate the total cost contribution of each chain. The chain with the maximum cost is selected and the due date of the first activity in this chain is decreased by a single time unit. This due date value is used while scheduling the activity locally optimally, but the objective function is still calculated with the original problem data. By decreasing the due date of an activity, other members of the chain can move earlier in time and the objective function value may be improved. If this is the case, then we identify an improved schedule associated with the current activity list. The procedure is repeated for other chains in non-increasing order of their contributions to the objective function. The search is terminated, if either the prespecified maximum number of non-improving steps or the maximum number of neighborhood search steps is reached first. .

3.5. LP-Based Optimal Timing

A final improvement step is applied to a limited number of activity lists in the initial population. We insert additional arcs into the precedence graph which avoid resource infeasibilities based on the current feasible schedule associated with the activity list (e.g., see Artigues and Roubellat, 2000). This allows us to formulate an LP which yields a resource-feasible optimal schedule for the given extended precedence graph. In the LP formulation below, the set of extended precedence relationships $\bar{P}$ includes the original precedence relationships on top of the precedence relationships derived from the resource flows. In essence, the concept of resource flows allows us to convert conditions on resource feasibility into temporal

relationships. In our presentation, $(i,k,j,l) \in \bar{P}$ if there is either a precedence relationship or a resource flow between activities (i,k) and (j,l):

$$\min \sum_{i \in I} \sum_{j \in J_i} \left(e_{ij} \cdot E_{ij} + t_{ij} \cdot T_{ij} \right) + K \cdot d_{h+1}$$

$$f_{jl} - f_{ik} \geq p_{jl} \qquad \forall (i,k,j,l) \in \overline{P} \tag{12}$$

(4), (5), (6), (10), (11).

Note that the structure of the LP above is similar to the mathematical model of DRCMPSPWET, except that the binary variables x_{ijt} and the related constraints are replaced by constraints (12) under the presence of extended precedence relationships.

The number of activity lists to which the LP-based improvement step is applied is referrred to here as the number of LP-based search steps. The search for the best-performing values of this parameter together with the maximum number of neighborhood search steps and the maximum number of non-improving steps for both positional and neighborhood searches are the subject of the next section.

3.6. Parameter Fine-tuning

In order to select the best-performing parameter settings, a fine-tuning procedure is applied. 20 different instances with 200 activities are tested. Six different parameters are adjusted: the maximum number of steps for the positional neighborhood search, the maximum number of steps for the timing-based neighborhood search, two different parameters for the maximum number of non-improving steps of these neighborhoods, the number of LP-based search iterations, and the size of the tabu list in the positional neighborhood search. A preliminary analysis revealed that the solution quality and time are insensitive to the size of the tabu list. This parameter has therefore been fixed at 5 in the rest of our study. The different values selected for each setting and the results are presented in Table 1 and Figure 2, respectively.

Table 1: Parameter selection settings.

	Max # of Positional Neighborhood Search Steps	Max # of Timing-based Neighborhood Search Steps	Max # of Non-improving Steps for the Positional Neighborhood	Max # of Non-improving Steps for the Timing-Based Neighborhood	# of LP-Based Search Steps	Size of the Tabu List
Setting 1	20	30	10	20	5	5
Setting 2	50	50	20	40	5	5
Setting 3	100	100	30	70	10	5
Setting 4	200	200	100	150	10	5

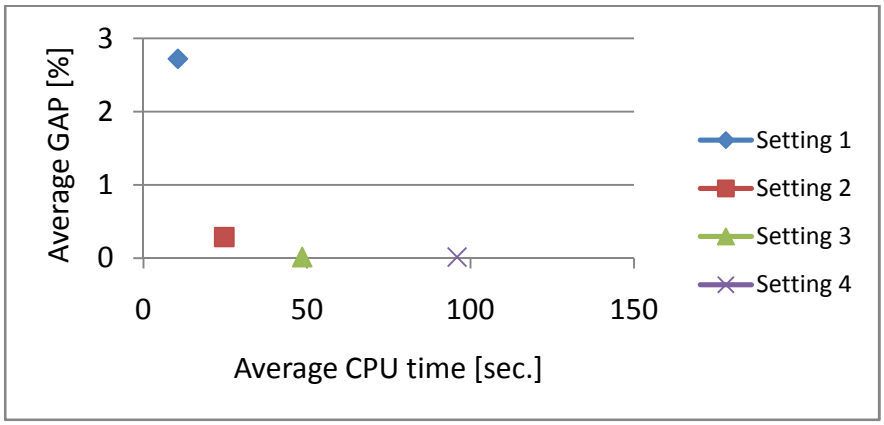

Figure 2: Parameter fine-tuning results.

Figure 2 shows the average gap between the best solution and the solution found by each setting and the average CPU times. Setting 3 attains the best trade-off in terms of solution quality and CPU times. Therefore, setting 3 is selected for the solution procedure(s) applied.

4. COMPUTATIONAL STUDY

All solution approaches were implemented in Visual C#. IBM ILOG CPLEX Optimization Studio 12.1 is used as the engine for solving the LP models. A data set of 800 unique instances is generated to test the performance of the suggested methods. The experiments were conducted on a single core of an HP Compaq DX 7400 Microtower with a 2.33 GHz Intel Core 2 Quad CPU Q8200 processor and 3.46 GB of RAM.

4.1. Experimental Data

As stated before, the related work in the literature focuses on the single project version of RCPSPWET. Moreover, existing benchmark instances do not always investigate the effects of different problem parameters on the performance of the proposed solution approaches. Therefore, a new data set is generated. Each instance of the problem set consists of a group of projects present in a baseline schedule with activity based due dates, unit earliness and tardiness costs. A newly arriving project is also included with a completion time factor K. The parameter settings for the entire data set are given in Table 2. The rationale behind adopting each of these parameters will be discussed in the upcoming subsections.

Table 2: Parameter settings for the data set generated.

Total Number of Activities	20, 40, 50, 100, 150, or 200
Due Date Range	Clustered or Distributed
Due Data Tightness	Tight or Loose

# of Resource Types	2 or 5
Completion Time Factor	High or Low

4.1.1. Project Pool Generation

Since our problem is a multi-project scheduling problem, each instance in the test problem data set consists of a group of projects. For this reason, a project pool is generated first, which will later be used to create the multi-project instances. Various random project generation procedures have been discussed in the literature. ProGen is developed by Kolisch *et al*. (1995) for RCPSP and its multi-mode extension. ProGen/max developed by Schwindt (1998) is an upgraded version of ProGen for minimal and maximal time lag extensions of generalized precedence relations. A more recent project generator, called RanGen, has been developed by Vanhoucke *et al.* (2003). We use this generator because RanGen enables the user to select predefined complexity measures for generated networks, which is important for differentiating the instances.

Four parameters have to be specified in RanGen to obtain different project networks. The first parameter is the order strength (OS), which is defined as the number of precedence relations including the transitive ones but not including those arcs incident from or into the dummy start and end activities, respectively, divided by the maximum number of precedence relations $n(n-1)/2$, where n denotes the number of non-dummy activities in the network (Mastor, 1970). RanGen is able to generate unique networks with the prespecified OS values. Three different OS values (0.25, 0.50, and 0.75) are selected. For each project, 5 types of renewable resources are defined. Two different resource usage related parameters are included. The first resource related measure is the resource density (RU) defined as below (13):

$$RU = RU_i = \sum_{r=1}^{R} \begin{cases} 1 & \text{if } w_{ir} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

This parameter specifies the number of resource types used by an activity *i*, RU_i, in the network. RU is preferred to another resource related measure referred to as the resource factor (RF) introduced by Alvarez-Valdes and Tamarit (1989), because RF might yield networks in which some activities do not use any resources at all. Another resource measure, the resource-constrainedness (RC), is defined as the ratio between the available capacity of a resource type (W_r) and the average usage of activities ($\overline{w}_r$) of this particular resource (14). The RU and RC values are selected as 4, 5, and 0.25, 0.50, respectively. The number of the activities in a project is taken as an input data as well. To achieve the required number of activities for each RCMPSPWET instance, projects with 5, 10, 20, and 30 activities are generated.

$$RC_i = \frac{\overline{w}_r}{W_r}; \tag{14}$$

Table 3: Settings for the project pool generation.

# of Activities	OS	RU	RC	# of Unique Projects
5	0.25	4	0.25	3
	0.50	4	0.50	10
	0.75	5	0.50	9
	0.50	5	0.25	10
10	0.25	4	0.25	10
	0.50	5	0.25	10
	0.75	5	0.25	10
	0.75	4	0.50	10
	0.25	5	0.50	10
20	0.25	5	0.25	10
	0.50	5	0.25	10
	0.75	5	0.25	10
	0.25	4	0.50	10
	0.75	4	0.50	10
30	0.25	5	0.25	10
	0.50	5	0.25	10
	0.75	5	0.25	10
	0.25	4	0.50	10
	0.75	4	0.50	10

All parameter settings are summarized in Table 3. The project pool for each *n*, except for $n = 5$, consists of 50 different projects. A total of 32 projects with 5 activities is used because that the generator is not able to generate 50 unique networks with the specified OS values due to the small number of nodes in the network.

The data set can be obtained by sending a request to the corresponding author.

4.1.2. Total Number of Activities

The total number of activities in an instance is an important measure of the size as well as the difficulty of the instance. As presented in Table 2, for a given instance the number of activities is ranging from 20 to 200 activities, excluding the dummy activities. Note that we solve instances with up to 200 activities while the maximum number of activities considered in the literature on earliness/tardiness project scheduling problems is 100 (Ballestin and Trautman (2008), Vanhoucke *et al.* (2001), and Neumann *et al*. (2003)).

4.1.3. Project Combinations

For each setting of the total number of activities, different combinations of projects are selected from the project pool to create an instance of DRCMPSPWET with the required number of activities. For example, in order to generate an instance of DRCMPSPWET with 30 activities, a combination of three projects with 10 activities each is selected as one of the combinations. In this scenario, for two of these three projects, due dates, earliness and tardiness costs are generated. The third project is defined as the newly arriving one, and a completion time factor is determined for it. Another combination uses a project portfolio of 6 projects with 5 activities, where one of these projects is designated as the new arrival. For each value of the total number of activities in Table 2, up to three different combinations are selected. These combinations differ in the total number of projects in an instance. For each combination, five different master instances are generated. These master instances provide the information about which projects in the pool are added to the project portfolio. This is accomplished by selecting projects from the pool with the desired number of activities randomly. Master instances are then used to create unique instances by adding the data about the due dates, the unit earliness and tardiness costs and the completion time factors depending on the values of the remaining data generation parameters. The unit earliness and tardiness costs are drawn from uniform distributions in the range 0 to 10, and the generation of the due dates and the completion time factors are detailed in Sections 4.1.4-4.1.6 and 4.1.8, respectively. All the project combination schemes are summarized in Table A in the Appendix.

4.1.4. Due Date Generation

Due dates are generated in this study is based on a baseline schedule. All projects in an instance, except for the new arrival, have an associated existing schedule constructed by the scheduling routine described next. In Ballestin and Trautman (2008), Vanhoucke *et al.* (2001), and Neumann *et al*. (2003), on the other hand, the data sets are generated by considering the critical paths and the earliest start time values of the activities in the network.

The method used to obtain the baseline schedule is quite important for the effective utilization of the resources. Therefore, makespan minimization is selected as the objective for generating the baseline schedule. There are many heuristic approaches in the literature developed for makespan minimization. We decided to use a scheduling scheme with an effective dispatching rule in order to generate schedules with good makespan values within reasonable computation times. In his review paper about the performance of different dispatching rules for makespan minimization, Kolisch (1996) states that the LST rule shows the best performance. Therefore, the LST rule is used here together with the serial scheduling scheme (SSS) for generating the baseline schedule. At each iteration, SSS selects the activity with the minimum LST among the ones whose predecessors are already scheduled and schedules it at the earliest feasible point in time leading to an active schedule. The LST values are calculated using the backward pass algorithm of the critical path method. However, we implement the LST rule slightly differently depending on the desired range of the due dates as discussed next.

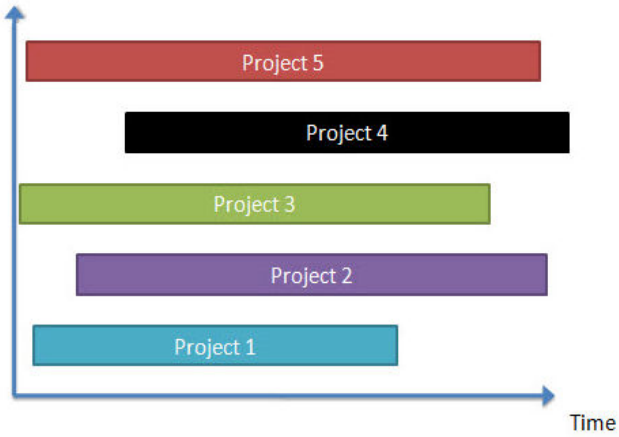

Figure 3: Distributed due date windows.

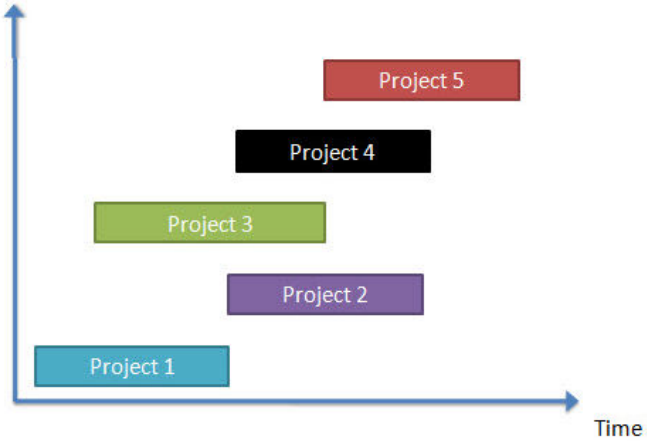

Figure 4: Clustered due date windows.

4.1.5. Due Date Range

The range of the due dates over the time horizon in the baseline schedule is important for flexibility in scheduling. That is, if the due dates of a project are spread over the entire planning horizon, activities can be moved forward or backward more freely in time while scheduling the arriving project. Clustered due dates, on the other hand, reduce the flexibility of the projects and constrain them to move only within shorter time windows provided in the baseline schedule. The difference between these two settings is visualized in Figures 3 and 4. In the first case, all projects are active during most of the schedule timeline whereas in the second case a relatively few projects are active within a given time interval.

In order to obtain schedules with these two different characteristics, the basic schedule generation scheme based on the LST rule is modified. For the distributed due date generation, we keep track of the progress levels of the projects while scheduling activities iteratively. In other words, the activity with the lowest LST value is picked among the activities of the project with the minimum progress level. With this approach, the projects are kept active along the entire timeline of the baseline schedule. The clustered due date range is obtained by randomly selecting a project and scheduling all activities of this particular project one by one according to the LST rule in order to complete the selected project as soon as possible after it is started. The process continues by selecting another unscheduled project randomly until all the projects are scheduled. In order to observe the effects of this parameter setting, distributed and clustered due date generation schemes are applied to project combinations with a relatively high number of projects. Otherwise, only the distributed due date generation scheme is employed. The details are provided in Table A in the Appendix.

4.1.6. Due Date Tightness

An additional parameter controls the tightness of the due dates. Tight due dates values are closer to the starting time of the schedule and offer less flexibility for meeting the due date. Loose due dates, on the other hand, allow delays to activities without affecting successor

activities or incurring additional cost. In other words, there is a higher possibility of meeting loose due dates compared to tight ones. Due date tightness in the related articles in the literature is manipulated by multiplying the individual due dates (or an average due date) by a tightness factor. We use a different approach. In order to reflect these tightness and looseness considerations in our data generation scheme, we change the available resource capacities in the baseline schedule. For setting loose due dates, we only allow a resource to be used at 80% of its available capacity. By creating a baseline schedule for the current project portfolio by utilizing the resources at less than full availability, slack resource capacities can be used to schedule a newly arriving project without causing significant deviations in the new schedule. As a consequence, the makespan of the baseline schedule is increased but additional resource capacity is allocated to schedule the newly arriving project. We would expect that the number of activities scheduled on time would increase and for the same master instance a lower objective value can be obtained. By a similar argument, a baseline schedule constructed with fully available resources would result in tight due dates. For each unique instance loose and tight due dates settings are present in the data set.

4.1.7. Number of Resources

The number of resource types is another complicating factor in a project scheduling problem. In general, instances with more resources are more challenging. In our data set, the number of resource types is either 2 or 5. Initially, all instances are created with five resource types, and the last three resource types are simply dropped from instances with 2 resource types.

4.1.8. Completion Time Factor

As one of the contributions of this study, the effects of the completion time factor K on the schedules will be studied, and we need to set a completion time factor value for each instance in the data set. As stated before, a trade-off between the earliness/tardiness costs of the activities in the baseline schedule and the completion time related cost of the new project must exist in order to obtain a reasonable problem setting. Otherwise, scheduling the newly arriving project at the beginning or at the end of the schedule, depending on the dominant cost component, might yield good solutions for most of the instances. Therefore, we implemented another pre-scheduling step, similar to that in the due date generation process, to obtain the completion time factors. For two due date range settings, different approaches are used. For the distributed due date setting, we generate a new schedule employing the LST rule after adding the new project to the set of projects present in the baseline schedule of the instance and then calculate the total earliness/tardiness cost for the projects in the baseline schedule. The completion time factor is obtained by dividing the total earliness/tardiness cost by the completion time of the newly arriving project. For the clustered due date range, recall that projects are added to the schedule one by one in some sequence. The new project is inserted into each possible position in this sequence, and the LST rule is invoked for the resulting order. We obtain different cost values for the same instance depending on the position of the new project in the sequence and then take the average of these cost values and also compute the

average completion time of the new project. The ratio of these two average values yields the completion time factor of the instance. Thus, we generated completion time factors specific to the instance data instead of selecting the same factor for all instances.

These completion time factor values are scaled in order to provide different parameter settings. In the "high" setting, the scaling factor is 1 and reflects that we expect the contribution of the new project to the overall objective function to be roughly the same as the total earliness/tardiness cost of the projects in the baseline schedule. In the "low" setting, the scaling constant is 0.5.

4.2. Results

We had two primary goals in mind while designing our computational study. First, we demonstrate that the local search method provides solutions of high quality in reasonable computation times. Second, we explore the effects of various problem parameters detailed in the previous sections on solution quality. We implemented two variants of our iterated local search algorithm as discussed in Section 3.2. In one variant (LS), the initial population consists of randomly generated lists in addition to activity lists produced by dispatch rules. In the second variant (LS-SB), the initial population is enhanced by activity lists retrieved from the SB heuristic mentioned in Section 3.2. Detailed results are available in Pamay (2011).

Table 4: Comparison of the local search method against CPLEX 12.1.

# of	# of	# of	Avg. Gap (%)			Max. Gap (%)			Avg. CPU Time (sec.)		
Act.	Res.	Inst.	MIP	LS	LS-SB	MIP	LS	LS-SB	MIP	LS	LS-SB
20	2	20	0.00	2.53	2.09	0.00	16.71	16.71	74	3	4
	5	20	0.00	5.21	0.15	0.00	35.05	1.28	476	6	27
30	2	40	0.10	9.78	8.74	3.39	40.10	40.10	1241	5	7
	5	40	1.55	5.05	3.24	15.07	22.91	20.21	4138	9	45
40	2	40	12.40	9.87	8.88	456.88	41.16	39.67	796	6	8
	5	40	7.87	10.60	7.74	55.85	70.40	59.20	5121	12	62
50	2	60		1.37	0.00		20.25	0.00		7	10
	5	60		4.77	0.00		56.14	0.00		14	57
100	2	80		0.49	0.00		28.10	0.00		14	20
	5	80		2.20	0.00		39.42	0.00		26	138
150	2	80		0.47	0.00		21.24	0.00		24	35
	5	80		1.20	0.00		57.13	0.00		46	231
200	2	80		0.04	0.00		3.40	0.00		35	49
	5	80		0.17	0.00		12.54	0.00		72	304

In the first part of our computational study, we benchmark the proposed local search method against the integer programming formulation (MIP) presented in Section 2.2 solved by

ILOG CPLEX 12.1 which can only handle instances with up to 40 activities. Larger instances require excessive computation times. The time limit imposed on CPLEX is one hour for instances with 20 and 30 activities, and two hours for instances with 40 activities. If CPLEX does not terminate with an optimal solution within the allotted time (39 and 42 instances with 30 and 40 activities, respectively), then we report the best integer solution identified during the optimization. Therefore, all gaps are computed with respect to the best solution available. The results in Table 4 are grouped by the number of activities and resource types (indicated in the first two columns), and the number of instances in the group is given in the third column. The results in Table 4 attest to the competitiveness of the iterated local search heuristic. All optimal solutions are available for 40 instances with 20 activities, where LS attains the optimal solution in 24 cases with an optimality gap of 3.87% on average. When the initial population is extended with activity lists from the SB heuristic, the number of optimal solutions identified increases to 29 with an average optimality gap of 1.12%. LS attains better solutions than MIP in 11 and 17 cases for instances with 30 and 40 instances, respectively. The corresponding numbers for LS-SB are 11 and 18. Both LS and LS-SB match the best solution obtained by MIP in 19 and 10 cases for instances with 30 and 40 activities, respectively. For instances with 40 activities, LS and MIP perform on a par, and LS-SB is superior to MIP; however, both LS and LS-SB take a fraction of the effort required by CPLEX. The diversification effect of the activity lists retrieved from the SB heuristic manifests itself in both the average and the maximum gaps. For instances with 50 or more activities the differences in the maximum gaps are particularly significant.

Table 5: Effect of the due date tightness on the solution quality.

# of	Due Date	# of	Avg. Gap (%)			Max. Gap (%)		
Activities	**Tightness**	**Instances**	**MIP**	**LS**	**LS-SB**	**MIP**	**LS**	**LS-SB**
20	Tight	20	0.00	4.22	1.35	0.00	35.05	16.71
	Loose	20	0.00	3.52	0.88	0.00	29.41	9.09
30	Tight	40	0.62	6.59	5.51	15.07	36.84	36.84
	Loose	40	1.04	8.25	6.47	14.55	40.10	40.10
40	Tight	40	14.41	12.59	10.66	456.88	53.34	44.42
	Loose	40	5.85	7.88	5.96	55.85	70.40	59.20
50	Tight	60		1.01	0.00		17.11	0.00
	Loose	60		5.12	0.00		56.14	0.00
100	Tight	80		1.06	0.00		28.10	0.00
	Loose	80		1.63	0.00		39.42	0.00
150	Tight	80		1.45	0.00		57.13	0.00
	Loose	80		0.22	0.00		14.56	0.00
200	Tight	80		0.00	0.00		0.00	0.00
	Loose	80		0.21	0.00		12.54	0.00

It is not possible to identify a uniform pattern regarding the effect of the due date tightness on the solution quality from the data in Table 5. The results for instances with 40 or

less activities suggest that instances with loose due dates are somewhat easier.

Next, we investigate the impact of the distributed and clustered due dates on the iterated local search heuristic. Recall that instances with up to 40 activities are all generated with the "distributed" option; therefore, no MIP result is available for this analysis. Results presented in Table 6 suggest that the added value of the extended initial population is more critical when the due dates are distributed.

Table 6: Effects of the due date range on the solution quality.

# of Activities	Due Date Range	# of Instances	Avg. Gap (%)		Max. Gap (%)	
			LS	LS-SB	LS	LS-SB
50	distributed	80	4.27	0.00	56.14	0.00
	clustered	40	0.65	0.00	12.50	0.00
100	distributed	120	1.68	0.00	39.42	0.00
	clustered	40	0.37	0.00	8.80	0.00
150	distributed	120	0.16	0.00	14.56	0.00
	clustered	40	2.87	0.00	57.13	0.00
200	distributed	120	0.14	0.00	12.54	0.00
	clustered	40	0.00	0.00	0.00	0.00

Table 7: Effects of the completion time factor on the solution quality.

# of Activities	Completion Time Factor	# of Instances	Avg. Gap (%)			Max. Gap (%)		
			MIP	LS	LS-SB	MIP	LS	LS-SB
20	Low	20	0.00	5.07	1.89	0.00	35.05	16.71
	High	20	0.00	2.67	0.34	0.00	29.41	5.00
30	Low	40	1.58	7.61	5.43	15.07	36.84	36.84
	High	40	0.08	7.23	6.55	1.25	40.10	40.10
40	Low	40	5.09	10.47	8.02	55.85	53.34	38.42
	High	40	15.18	10.00	8.60	456.88	70.40	59.20
50	Low	60		3.86	0.00		56.14	0.00
	High	60		2.27	0.00		42.40	0.00
100	Low	80		2.23	0.00		39.42	0.00
	High	80		0.47	0.00		13.76	0.00
150	Low	80		0.44	0.00		16.08	0.00
	High	80		1.23	0.00		57.13	0.00
200	Low	80		0.21	0.00		12.54	0.00
	High	80		0.00	0.00		0.00	0.00

Finally, Table 7 explores the sensitivity of our results to the completion time factor of the new project. It is evident that LS and LS-SB return solutions of high quality under both the “low” and “high” settings of the completion time factor. The effect of the extended initial population is more pronunced for smaller values of the completion time factor.

In summary, the proposed iterated local search heuristic delivers solutions of high quality. Instances with up to 200 activities are solved in short CPU times given that our problem is not an operational problem and does not need to be solved frequently. Furthermore, the performance of our algorithm is robust under various data generation settings; in particular, if we opt for using an enhanced initial population as described in Section 3.2.

5. CONCLUDING REMARKS AND FUTURE WORK

The purpose of this work is to study the dynamic project scheduling environments. In that problem setting a project arrives on top of an existing project portfolio and a due date has to be quoted for the new project while keeping the costs related to changes in the schedule at a minimum. The objective function consists of the weighted earliness tardiness costs of the activities of the existing projects in the current schedule in addition to a term that increases linearly with the anticipated completion time of the new project. An iterated local search heuristic is developed to solve large instances of this problem. In order to analyze the performance of the proposed method, a new multi-project data set is created by controlling the due date tightness, the due date range, the number of resource types, the completion time factor, and the total number of activities in an instance. A series of computational experiments are carried out to test the performance of the local search approach. Moreover, exact solutions for the small instances are provided. The results indicate that the proposed local search heuristic performs well in terms of both solution quality and solution time. The value of an extended initial population is also demonstrated.

Several interesting extensions of this work are listed below.

- Precedence relations between projects can also be included considering that in practice some projects need to precede others due to technological factors, e.g., in R&D environments.
- Arrival of multiple projects at a time or at different points in time may be studied.
- A multi-mode extension is clearly an important research direction we may pursue in the future.

To the best of our knowledge, the proposed work is the first study of the multi-project dynamic version of RCPSPWET, namely, DRCMPSPWET. The relative scarcity of the literature on this problem suggests that static and dynamic resource constrained multi-project scheduling problems with weighted earliness tardiness costs constitute a rich topic for further research activities. Moreover, the practical relevance of this problem for companies, which have to manage their project portfolio in dynamic environments, offers a wide range of implementation options in the business context.

Acknowledgments: We gratefully acknowledge the support given by The Scientific and Technological Research Council of Turkey (TUBITAK) through Project Number MAG 109M571.

REFERENCES

J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, 34, 391-401, 1988.

R. Alvarez-Valdes and J. M. Tamarit, "Heuristic algorithms for resource constrained project scheduling: A review and empirical analysis", in Slowinski and Weglarz (Editors), *Advances in Project Scheduling*, Elsevier, The Netherlands, pp. 113-134, 1989.

C. Artigues and F. Roubellat, "A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes," *European Journal of Operational Research*, 127, 2, pp. 297 – 316, 2000.

B. Ashtiani, R. Leus, and M. Aryanezhad, "New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing," *Journal of Scheduling*, 14, pp. 157-171, 2011.

F. Ballestin and N. Trautman, "An iterated-local-search heuristic for the resource-constrained weighted earliness-tardiness project scheduling problem," *International Journal of Production Research*, 46, pp. 6231–6249, 2008.

A.B. Badiru and P.S. Pulat, *Comprehensive Project Management*, Prentice Hall PTR, New Jersey, 1995.

J. Blazewicz, J. Lenstra, and A. Rinnooy Kan, "Scheduling subject to resource constraints - classification and complexity," *Discrete Applied Mathematics*, 5, pp. 11–24, 1983.

K. Bulbul and P. Kaminsky, "A Linear Programming-Based General Method for Job Shop Scheduling," *Journal of Scheduling*, in press, 2010. http://dx.doi.org/10.1007/s10951-012-0270-4.

E. Demeulemeester, and W. Herroelen, *Project Scheduling. A Research Handbook*, Kluwer Academic Publishers Group, Dordrecht, The Netherlands, 2002.

E. Demirkol, S. Mehta, and R. Uzsoy, "A computational study of shifting bottleneck procedures for shop scheduling problems," *Journal of Heuristics*, 3(2), pp. 111–137, 1997.

H. El Sakkout and M. Wallace, "Probe backtrack search for minimal perturbation in dynamic scheduling," *Constraints,* 5, 4, pp. 359–388, 2000.

J. Herbots, W. Herroelen, and R. Leus, "Dynamic order acceptance and capacity planning on a single bottleneck resource," *Naval Research Logistics*, 54, 8, pp. 874–889, 2007.

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

J. Kanet and V. Sridharan, "Scheduling with inserted idle time: problem taxonomy and literature review," *Operations Research*, 48(1), pp. 99-110, 2000.

R. Kolisch, "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation," *European Journal of Operational Research*, 90, 2, pp. 320 – 333, 1996.

R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problems," *Management Science*, 41, pp. 1693–

1703, 1995.
R. Kurtulus and E. W. Davis, “Multi-project scheduling: categorization of heuristic rules performance,” *Management Science*, 28, pp. 161–172, 1982.
J. Lenstra, A. Rinnooy Kan, and P. Brucker, “Complexity of machine scheduling problems,” *Annals of Discrete Mathematics*, 1, pp. 343–362, 1977.
S. Mason, J. Fowler, and W. Carlyle, “A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops,” *Journal of Scheduling*, 5(3), pp. 247–262, 2002.
A. Mastor, “An experimental investigation and comparative evaluation of production line balancing techniques,” *Management Science*, 16, 11, pp. 728–746, 1970.
K. Nanobe and T. Ibaraki, “A metaheuristic approach to the resource constrained project scheduling with variable activity durations and convex cost functions”, in Jozefowska and Weglarz (Editors) *Perspectives in Modern Project Scheduling*, International Series in Operations Research & Management Science, 92, Springer, Berlin, pp. 225-248, 2006.
K. Neumann, C. Schwindt, and J. Zimmermann, *Project Scheduling with Time Windows and Scarce Resources*, 2nd ed., Springer Verlag, Berlin, 2003.
M. B. Pamay, “A linear programming based method for the resource constrained multi-project scheduling problem with weighted earliness/tardiness costs,” MSc thesis, Sabanci University, Turkey, 2011. http://research.sabanciuniv.edu/17694/.
J. H. Payne, “Management of multiple simultaneous projects: A state-of-the-art review,” *International Journal of Project Management*, 13, pp. 163–168, 1995.
M. Pinedo, and M. Singer, “A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop,” *Naval Research Logistics*, 46(1), pp. 1–17, 1999.
M. Singer, “Decomposition methods for large job shops,” *Computers & Operations Research*, 28(3), pp. 193–207, 2001.
M. Vanhoucke, E. Demeulemeester, and W. Herroelen, “An exact procedure for the unconstrained weighted earliness tardiness project scheduling problem,” Research Report 9907, Department of Applied Economics, Katholieke Universiteit Leuven, no. 9907, 1999.
M. Vanhoucke, E. Demeulemeester, and W. Herroelen, “An exact procedure for the resource-constrained weighted earliness tardiness project scheduling problem,” *Annals of Operations Research*, 102, pp. 179–196, 2001.
M. Vanhoucke, “Optimal due date assignment in project scheduling,” Working Paper, Ghent University and Vleric Luevent Gent Management School, no. 159, 2002.
M. Vanhoucke, E. Demeulemeester, and W. Herroelen, “RanGen: A random network generator for activity-on-the-node networks,” Tech. Rep., Department of Applied Economics, Katholieke Universiteit Leuven, 2003.
K. K. Yang and C. Sum, “An evaluation of due date, resource allocation, project release, and activity scheduling rules in a multi-project environment,” *European Journal of Operational Research*, 13, pp. 139–154, 1997.

Appendix

Table A: Details of the data set generated for the computational study.

# of Activities	Combinations	ID	# of MI	Due Date Range	Due Date	# of Resource Types	*K*	E/T Cost Values	# of Instances
20	(5 A x 3P + 5A x 1P)	A20_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
30	(10 A x 2P + 10 A x 1P)	A30_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
30	(5 A x 5P + 5 A x 1P)	A30_2	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
40	(10 A x 3P + 10 A x 1P)	A40_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
40	(10 A x 3P 5A x 1P + 5 A x 1P)	A40_2	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
50	(10 A x 4P + 10 A x 1P)	A50_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
50	(5A x 8P + 10 A x 1P)	A50_2	5	Clustered or Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	80
100	(30 A x 3P + 10 A x 1P)	A100_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
100	(10 A x 2P 20A x 3P + 20 A x 1P)	A100_2	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
100	(10 A x 9P + 10 A x 1P)	A100_3	5	Clustered or Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	80
150	(30 A x 4P + 30 A x 1P)	A150_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
150	(20 A x 6P + 30 A x 1P)	A150_2	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
150	(10 A x 10P 30 A x 1P+ 20 A x 1P)	A150_3	5	Clustered or Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	80
200	(30 A x 6P + 20 A x 1P)	A200_1	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
200	(20 A x 8P 10A x 1P+ 30 A x 1P)	A200_2	5	Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	40
200	(5 A x 10P 10A x 12P+ 30 A x 1P)	A200_3	5	Clustered or Distributed	Loose or Tight	2 or 5	High or Low	U(0,10)	80