Journal of Heuristics manuscript No.
(will be inserted by the editor)

A Lagrangian Relaxation and ACO Hybrid for Resource Constrained Project Scheduling with Discounted Cash Flows

Dhananjay Thiruvady · Mark Wallace · Hanyu Gu · Andreas Schutt

the date of receipt and acceptance should be inserted later

Abstract We consider a project scheduling problem where a number of tasks need to be scheduled. The tasks share resources, satisfy precedences, and all tasks must be completed by a common deadline. Each task is associated with a cash flow (positive or negative value) from which a "net present value" is computed dependent upon its completion time. The objective is to maximize the cumulative net present value of all tasks. We investigate (1) Lagrangian relaxation methods based on list scheduling, (2) ant colony optimization and hybrids of (1) and (2) on benchmark datasets consisting of up to 120 tasks. Considering lower bounds, i.e., maximizing the net present value, the individual methods prove to be effective but are outperformed by the hybrid method. This difference is accentuated when the integrality gaps are large.

1 Introduction

Project scheduling has been a topic of interest for several decades. Typically, a project consists of a number of tasks and the aim is to optimize an objective

Dhananjay Thiruvady
Clayton School of IT, Monash University & CSIRO Computational Informatics, Victoria 3169, Australia
Tel.: +61 3 99058474
E-mail: dhananjay.thiruvady@csiro.au

Mark Wallace
Caulfield School of IT, Monash University, Victoria 3800, Australia
E-mail: mark.wallace@monash.edu

Hanyu Gu
School of Mathematical Sciences, University of Technology, Sydney, 15 Broadway, Ultimo NSW 2007, Australia
E-mail: Hanyu.Gu@uts.edu.au

Andreas Schutt
National ICT Australia, The University of Melbourne, Victoria 3010, Australia
E-mail: andreas.schutt@nicta.com.au

related to the completion time/value of the tasks. In previous research, the objective has mainly been to minimize a project duration but more recently there has also been interest in maximizing a net present value (NPV) of the project, which is a cash flow dependent on the task completion date (Chen et al, 2010; Vanhoucke, 2010; Show, 2006).

Project scheduling has been formulated in various ways (Brucker et al, 1999; Demeulemeester and Herroelen, 2002; Neumann et al, 2003). Brucker et al (1999) provides an overview of a number of variants. They focus on projects consisting of tasks, shared (renewable) resources, precedences between tasks and deadlines. Amongst methods, they discuss branch & bound, heuristic and local search approaches. Demeulemeester and Herroelen (2002) also describes a variety of problems from this class and detail ways in which these problems have been tackled. These include various exact methods, heuristic schemes and meta-heuristics including genetic algorithms, simulated annealing and tabu search. Neumann et al (2003) examine project scheduling with time windows also detailing various exact and heuristic methods.

Project scheduling is closely related to resource constrained job scheduling (Singh and Ernst, 2011; Thiruvady et al, 2012, 2014). The problem considered by these studies have similar characteristics (e.g. precedences and resource constraints between jobs, deadlines, etc.) with the main difference being the objective which is to minimise the total weighted tardiness. Singh and Ernst (2011) examine a Lagrangian relaxation based heuristic which proves to be more effective than heuristics on their own. Thiruvady et al (2012, 2014) explore ant colony optimisation and hybrids with constraint programming for a similar problem with hard deadlines.

Various studies have investigated methods (heuristic and exact) for the NPV problem (Chen et al, 2010; Vanhoucke, 2010; Show, 2006; Kimms, 2001; Gu et al, 2013). Chen et al (2010) investigate an ant colony optimization approach and show that their method outperforms other heuristic methods based on genetic algorithms, simulated annealing and tabu search for instances with up to 98 tasks. Vanhoucke (2010) also considers a heuristic scatter search approach and shows that this method is more effective than a previously suggested branch & bound approach for the same problem (Vanhoucke et al, 2001). Gu et al (2013) investigate a Lagrangian relaxation and constraint programming hybrid for the same problem and show that improved good feasible solution can be obtained with this approach. Show (2006) also investigate ant colony optimization for a similar problem with up to 50 tasks.

Lagrangian relaxation (LR) is a well-known technique applied to integer programming problems (Fisher, 2004). Many computationally hard problems can be tackled by considering a simpler version of the problem which omits (or 'relaxes') some complicating constraints. The solution to the relaxed problem can provide useful information about the original problem. In particular, Lagrangian relaxation methods provide an alternative way to obtain upper bounds (for maximization problems) providing performance guarantees. This is done by adding a cost term to the objective which is negative when any of the relaxed constraints would have been violated. Since the objective is

to maximize, this cost drives the solution towards satisfying the constraints. In the context of a NPV-based problem, Kimms (2001) showed how such a scheme could be successful.

Ant colony optimization (ACO) is a reinforcement-based meta-heuristic based on the foraging behavior of real ants which has been successfully applied to various combinatorial optimization problems (Dorigo and Stűtzle, 2004). This includes variants project scheduling problems (Merkle et al, 2000; Chen et al, 2010; Show, 2006). Merkle et al (2000) investigate the resource constrained project scheduling using makespan as the objective. Chen et al (2010) consider a problem where tasks can be executed in multiple modes. Show (2006) also consider a similar problem to Vanhoucke (2010).

We investigate the resource constrained project (RCP) scheduling problem suggested by Kimms (2001). The problem consists of maximizing the NPV of all the tasks subject to precedences between some tasks and a common deadline for all the tasks. Furthermore, all the tasks may require a number of renewable resources. There is a maximum availability on each of these resources for every time point. The previous study by Kimms (2001) develops a Lagrangian relaxation based heuristic and show that this approach is effective at obtaining tight upper bounds. We further extend these results here to show that lower bounds can be improved with the assistance of ACO. LR and ACO can effectively be applied to the RCP problem independently. However, here we show that the hybrid of these methods, LR-ACO, proves to be the most effective method to obtain lower bounds outperforming LR and ACO individually.

This paper is organized as follows. The RCP problem is stated formally in Section 2. Section 4 describes ACO and how it has been tailored for the current problem. In Section 5, the experimental details and an analysis of the results are provided. Section 7 concludes the paper.

2 Problem Formulation

The RCP scheduling problem can formally be stated as follows. There are a number of tasks $T = \{o_1, \ldots, o_n\}$ with each task consisting of a duration $d_i, i \in T$. During the execution of a task, there are associated cash flows. Let cf_{it} be the cash flow of task i in period t. The total cash flow of a task c_i can be computed as $\sum_{t=1}^{d_i} cf_{it} e^{\alpha(d_i - t)}$ where α is a discount rate. The discounted value of the task at the beginning of the project can be computed as $c_i e^{-\alpha(s_i + d_i)}$ where s_i is the start time of the task. Precedences between tasks may exist and are denoted by the set $P = \{(i_1, j_1), \ldots, (i_m, j_m)\}, i, j \in T$.

The constraints to be satisfied include resource and deadline constraints. Given k resources $R = \{R_1, \ldots, R_k\}$ with constant availability, each task requires r_{ik} units of the k^{th} resource. Additionally, every task must be completed by a pre-defined deadline δ.

The objective is to maximize the NPV

$$max. \quad \sum_{i=1}^{n} c_i e^{-\alpha(s_i+d_i)} \tag{1}$$

$$s.t. \quad s_i + d_i \leq s_j \qquad \forall (i,j) \in P \tag{2}$$

$$\sum_{i \in S(t)} r_{ik} \leq R_k \qquad k = \{1,\ldots,m\}, \quad t = \{1,\ldots,\delta\} \tag{3}$$

$$s_i + d_i \leq \delta \qquad i \in T \tag{4}$$

where $S(t)$ is the set of tasks executing at time t. The objective, Equation 1, is the net present value which is to be maximized. Constraint 2 specifies all tasks must start after their predecessors have completed. Constraint 3 requires that all the resources are satisfied and the final constraint requires that all the deadlines are satisfied.

An alternative way to view a solution is by considering permutation of tasks π. Now, a scheduling scheme can take π and map it to a resource-feasible schedule and let $\sigma(\pi)$ be such a mapping. Then, a feasible schedule is one that assigns start times to all tasks by satisfying precedence and resource constraints. This schedule has a NPV associated which each task and the cumulative NPV can be determined trivially and is represented as $f(\sigma(\pi))$. The precedences may be accounted for within a permutation, i.e., if a task i precedes task j then i appears earlier in the permutation then j. An alternative is to allow a preceding task to appear later than its successor in the permutation and modify the scheduling scheme σ to nevertheless generate a feasible schedule from it in which task i precedes task j. We found the latter scheme more effective and we therefore use this scheme in this study.

Permutations may not necessarily map to feasible schedules in terms of satisfying the deadlines. However, this situation is avoided by specifying sufficiently large deadlines such that feasible solutions are easily found. Hence, constraint (4) is redundant. Since the aim in this study is not to minimize makespan there is no issue with specifying large deadlines except in the situation with negative-valued cash flow tasks which may be scheduled later with larger deadlines.

3 Lagrangian Relaxation

In this study we consider two integer programming (IP) models to implement the LR method. The first is the one proposed by (Kimms, 2001) and the second is an adaptation of a similar model used by Singh and Ernst (2011) which was originally used for a resource constrained job scheduling problem.[1] The motivation for using the latter model is that it is a stronger formulation providing improved run-time scalability.

[1] This problem consists of a single renewable resource, tasks with release, processing and due times and the objective is to minimize the total weighted tardiness. See Singh and Ernst (2011) for further details.

3.1 LR-Kimms

We provide the IP model suggested by Kimms (2001) and briefly describe the Lagrangian function.[2] Binary variables x_{jt} are defined such that $x_{jt} = 1$ if task j completes at time t and 0 otherwise. The problem can be specified as follows.

$$max. \quad \sum_{i=1}^{n}\sum_{t=1}^{\delta} c_i e^{-\alpha t} x_{it} \tag{5}$$

$$s.t. \quad \sum_{t=1}^{\delta} x_{it} = 1 \qquad i \in T \tag{6}$$

$$\sum_{t=1}^{\delta} t x_{jt} - \sum_{t=1}^{\delta} t x_{it} \geq d_j \qquad \forall (i,j) \in P \tag{7}$$

$$\sum_{i=1}^{n} \sum_{\hat{t}=t}^{t+d_i-1} r_{ik} x_{i\hat{t}} \leq R_{kt} \qquad k \in R, \quad t \in \{1,\ldots,\delta\} \tag{8}$$

$$x_{it} \in \{0,1\} \qquad i \in T, \quad t \in \{1,\ldots,\delta\} \tag{9}$$

Equation 6 requires that all tasks complete. The precedences are incorporated via Equation 7 and the resource constraints via Equation 8.

The Lagrangian relaxation of the above problem can be obtained by relaxing the resource constraints and introducing multipliers λ_{kt}, $k \in R, t \in \{1,\ldots,\delta\}$. An upper bound can be obtained by solving the Lagrangian dual

$$LRR(\lambda) = max. \quad \sum_{i=1}^{n}\sum_{t=1}^{\delta} c_i e^{-\alpha t} x_{it} + \sum_{k\in R}\sum_{t=1}^{\delta} \lambda_{kt} \left(R_{kt} - \sum_{i=1}^{n} \sum_{\hat{t}=t}^{t+d_j-1} r_{i\hat{t}} x_{i\hat{t}} \right) \tag{10}$$

Subject to Equations 6,7 and 9. The above objective can be rearranged to obtain

$$LRR(\lambda) = max. \quad \sum_{i=1}^{n}\sum_{t=1}^{\delta} x_{it} \left(c_i e^{-\alpha t} - \sum_{i=1}^{n} \sum_{\hat{t}=t-d_j+1}^{t} \lambda_{k\hat{t}} r_{i\hat{t}} \right) + \left(\sum_{k\in R}\sum_{t=1}^{\delta} \lambda_{kt} R_{kt} \right) \tag{11}$$

where the last term is a constant and can be ignored when optimizing.

[2] For complete details please refer to Kimms (2001).

3.2 LR-SE

The second model is adapted from the one suggested by Singh and Ernst (2011) for multiple resources. As above, binary variables x_{it} represent the completion time of a task. However, unlike LR-Kimms, once a task completes it stays completed (see Equation 13 below).

$$max. \quad \sum_{i=1}^{n}\sum_{t=2}^{\delta} c_i e^{-\alpha t}(x_{it} - x_{it-1}) \tag{12}$$

$$s.t. \quad x_{it} \geq x_{it-1} \qquad i \in T, \quad t \in \{1,\ldots,\delta\} \tag{13}$$

$$x_{i\delta} = 1 \qquad i \in T \tag{14}$$

$$x_{jt} \leq x_{it-d_j} \qquad (i,j) \in P, \quad t \in \{1,\ldots,\delta\} \tag{15}$$

$$\sum_{i=1}^{n} r_{it}(x_{it} - x_{it-d_i}) \leq R_{kt} \qquad k \in R, \quad t \in \{1,\ldots,\delta\} \tag{16}$$

$$x_{it} \in \{0,1\} \qquad i \in T, \quad t \in \{1,\ldots,\delta\} \tag{17}$$

Equation 13 enforces that a task stays completed once it has finished and Equation 14 requires that all tasks complete. The precedences are enforced via Equation 15 and the resources constraints are specified by Equation 16.

As above, Lagrangian multipliers λ_{kt}, $k \in R, t \in \{1,\ldots,\delta\}$ are introduced and an upper bound can be obtained by solving the Lagrangian dual

$$\begin{aligned} LRR(\lambda) = max. \quad & \sum_{i=1}^{n}\sum_{t=2}^{\delta} c_i e^{-\alpha t}(x_{it} - x_{it-1}) + \\ & \sum_{k\in R}\sum_{t=1}^{\delta} \lambda_{kt}\left(R_{kt} - \sum_{i=1}^{n}\sum_{\hat{t}=t}^{t+d_j-1} r_{i\hat{t}}(x_{i\hat{t}} - x_{i\hat{t}-d_i})\right) \end{aligned} \tag{18}$$

which can be rearranged to obtain

$$\begin{aligned} LRR(\lambda) = max. \quad & \sum_{i=1}^{n}\sum_{t=2}^{\delta}(x_{it} - x_{it-1})\left(c_i e^{-\alpha t} - \sum_{i=1}^{n}\sum_{\hat{t}=t-d_j+1}^{t} \lambda_{kt} r_{i\hat{t}}\right) + \\ & \left(\sum_{k\in R}\sum_{t=1}^{\delta} \lambda_{kt} R_{kt}\right) \end{aligned} \tag{19}$$

with the last term being constant.

Algorithm 1 LR for the RCP problem

1: INPUT: An RCP instance
2: π^{bs} := NULL (best solution)
3: initialize $\lambda^0_{kt} = 0, \forall k \in R, \forall t \in \{1, \ldots, \delta\}$
4: $\gamma := 2.0$, $k := 0$, $gap := \infty$, $UB^* := \infty$, $LB^* := -\infty$
5: **while** $\gamma > 0.01$ & $gap > 0.01$ & $i < 1000$ **do**
6: ST = Solve(λ^i, UB)
7: π = GenerateList(ST)
8: ImproveLB(π)
9: UpdateBest(π^{bs},π,γ)
10: $LB* = NPV(\pi^{bs})$
11: UpdateMult(λ^i, LB^*, UB^*, ST, γ)
12: $gap = \frac{|UB^*|-|LB^*|}{|UB^*|}$
13: i ← i + 1
14: **end while**
15: OUTPUT: π^{bs}

3.3 The Lagrangian Relaxation Heuristic

Given both models, the high-level LR algorithm is presented in Algorithm 1. To begin with, various parameters and the multipliers are initialized. The main loop starts at line 5 and executes for 1000 iterations. The gap is above a specified threshold and $\gamma = 2.0$ is a scaling factor.[3] Each procedure is described in detail below.

Solve(λ^i, UB): The relaxed problem is solved within this procedure. This involves solving Lagrangian function, i.e., Equation 11 or Equation 20 depending on the model being used. The upper bound UB is set to

$$UB = LLR(\lambda^i) \tag{20}$$

which is minimized to provide tight upper bounds. The procedure returns a set of start times for the tasks (ST) representing the optimal relaxed solution.

GenerateList(ST): The start times obtained are transformed into a list of tasks. This is done by selecting the earliest start time, appending the corresponding task to π and continuing in the same way with the remaining tasks. In case of ties, tasks with higher NPV values are chosen first. The procedure returns a complete list of tasks (π).

ImproveLB(π): π can be mapped to a feasible schedule as will be seen in Section 4.1. This provides a lower bound to the optimal NPV. However, this lower bound may be improved further with the assistance of an alternative method resulting in a modified permutation π. The hybrid method is obtained by using ACO here.

[3] γ is progressively decreased to ensure the algorithm converges.

UpdateBest(π^{bs},π,γ): $\pi^{bs} = \pi$ if $f(\sigma(\pi)) > f(\sigma(\pi^{bs}))$. Additionally, if π^{bs} has not been updated in the last five iterations, $\gamma \leftarrow \gamma \div 2$.

UpdateMult(λ^i, LB^*, UB^*, ST, γ): The multipliers are updated for all time periods $t \in \{1, \ldots, \delta\}$, $k \in R$

$$\lambda_{k,t}^{i+1} = max\left(0, \lambda_{k,t}^{i} + \frac{\gamma(UB^* - LB^*)\Delta_{kt}}{\sum_{k\in R}\sum_{\hat{t}=1}^{\delta}\Delta_{k\hat{t}}^2}\right) \tag{21}$$

where in the case of LR-Kimms

$$\Delta_{kt} = \sum_{i=1}^{n}\sum_{\bar{t}=t}^{t+d_j-1} r_{ik}x_{i\bar{t}} - R_{kt} \tag{22}$$

The above equation can be modified for the LR-SE model as follows

$$\Delta_{kt} = \sum_{i=1}^{n}\sum_{\bar{t}=t}^{t+d_j-1} r_{i\bar{t}}(x_{i\bar{t}} - x_{i\bar{t}-d_i}) - R_{kt} \tag{23}$$

4 Ant Colony Optimization

ACO was first suggested by Dorigo (1992) for combinatorial optimization. When looking for food, ants will leave their nests and mark the paths that they use to a food source with pheromone. Other ants looking for food will follow these trails based on the amount of pheromone deposits on the paths. They, in turn, deposit pheromones on these paths. Thus, paths with more pheromone receive more ants which in turn deposit more pheromone leading to a positive feedback loop. This mechanism leads the colony to converge on better food sources over time (Camazine et al, 2001).

In the context of the RCP problem, we consider a pheromone model that is based on learning an ideal permutation of the tasks. Here, the aim is to learn permutations of the tasks which are then mapped to a schedule using a scheduling scheme (discussed later). We consider the model suggested by den Besten et al (2000) who examine an ACO algorithm for the single machine problem with the total weight tardiness objective. The pheromones $\mathcal{T}$ consist of pheromone values τ_{ij} for each task j and variable i or position in the sequence. The motivation to use this model is that in the absence of any obvious dependencies[4] selecting a task for a variable is the simplest model.

Two popular variants of ACO are ant colony system (Dorigo and Gambardella, 1997) (ACS) and Max-Min ant system (Stűtzle and Hoos, 2000). Initial experiments with both variants showed that ACS was better suited to this problem. Thus, for all experiments in this study we used ACS.

[4] For example, in the travelling salesman problem selecting a city based on the next one is important (Dorigo and Gambardella, 1997) and hence t_{ij} represents the desirability of selecting city j given that city i was the previously selected city.

Algorithm 2 ACS for the RCP problem

1: INPUT: An RCP instance
2: π^{bs} := NULL (global best)
3: initialize $\mathcal{T}$
4: π^{bs} = ACSImproveLb($\mathcal{T}$)
5: OUTPUT: π^{bs}

Algorithm 3 ACSImproveLb

1: INPUT: $\mathcal{T}$
2: π^{bs} := NULL (best)
3: **while** termination conditions not satisfied **do**
4: $S_{iter} := \emptyset$
5: **for** $j = 1$ to n_a **do**
6: π_j := ConstructSolution($\mathcal{T}$)
7: ScheduleTasks(π_j)
8: $S_{iter} := S_{iter} \cup \{\pi_j\}$
9: **end for**
10: $\pi^{ib} := \mathsf{argmin}\{f(\pi)|\pi \in S_{iter}\}$
11: Update(π^{ib},π^{bs})
12: PheromoneUpdate($\mathcal{T}$,π^{bs})
13: cf := ComputeConvergence(π^{ib})
14: **if** $cf = true$ **then** initialize $\mathcal{T}$ **end if**
15: **end while**
16: OUTPUT: π^{bs}

The ACS algorithm is presented in Algorithm 2 and Algorithm 3. In Algorithm 2 we see the high level ACO procedure. The global best ant (π^{bs}) and the pheromone trails $\mathcal{T}(\tau_{ij} = 1/n, \forall i, j)$ are first initialised. Now algorithm 3 is called with $\mathcal{T}$ as the input. A best solution (π^{bs}) is also maintained by this algorithm and runs until some terminating criteria is met (line 3) such as number of iterations or time limit. Within each iteration, n_a ants construct permutations (line 6) which are mapped to schedules (line 7).

ConstructSolution($\mathcal{T}$): A permutation π of tasks is constructed by selecting a task for each variable[5] starting at π_1. A complete solution obtains a permutation where all variables have unique tasks assigned to them. A task is selected with one of two schemes, either deterministic or probabilistic. First, a random number $q \in (0, 1]$ is generated and compared with a pre-defined parameter q_0 in order to select a task at π_i. If $q < q_0$, k is deterministically selected according to

$$k = max_{k \in \mathcal{J} \setminus \{\pi_1, \ldots, \pi_{i-1}\}} \{\tau_{ik} \eta_k^{\beta}\} \tag{24}$$

[5] π consists of n variables where each variable is to be assigned to a task. We also tested the sum rule often used in scheduling applications (Dorigo and Stűtzle, 2004) but found no advantage using this method but was slightly worse overall (see Appendix A).

otherwise, k is probabilistically selected from the following distribution

$$\mathbf{P}(\pi_i = k) = \frac{\tau_{ik}\eta_k^\beta}{\sum_{j \in \mathcal{J} \setminus \{\pi_1,\ldots,\pi_{i-1}\}} \left(\tau_{ij}\eta_j^\beta\right)} \tag{25}$$

where η_k is heuristic information that may be used to bias the search and β is a factor that determines the contribution of the heuristic information. We attempted various heuristics, such as favouring positive-valued cash flow tasks to be placed early in the sequence, but found no obvious advantage with any of them. Hence, $\beta = 0$ was used which effectively rules out heuristic information.

When a task j at variable i is selected, the pheromones are updated as follows:

$$\tau_{ij} = \tau_{ij} \times \rho + \tau_{min} \tag{26}$$

which is called a local pheromone update, where ρ is a learning rate parameter which is chosen to gradually reduce the levels of pheromone associated with task j at variable i. This favours diversity by allowing other tasks to be assigned to the same variable during future solution constructions. $\tau_{min} = 0.001$ is a lower limit which does not allow the probability of selection of a task to reduce to 0.

ScheduleTasks(): Once a sequence for the current solution has been specified the schedule $\sigma(\pi)$ is determined. This is done using a scheduling scheme (see Section 4.1) and depending on whether the permutation is precedence feasible the scheduling scheme satisfies precedences and resource constraints or only resource constraints. This scheme always generates resource feasible schedules given infinite time.

Update(π^{ib},π^{bs}): The procedure sets π^{bs} to π^{ib} if $f(\sigma(\pi^{ib})) > f(\sigma(\pi^{bs}))$ where $f(\sigma(\pi^{ib}))$ is the NPV of the iteration best solution.

PheromoneUpdate($\mathcal{T}$,π^{bs}): All components (i, j) appearing in π^{bs} are used to update the corresponding components in the pheromone matrix:

$$\tau_{ij} = \tau_{ij} \times \rho + \Delta \tag{27}$$

where $\Delta = 0.01$ is a reward amount set to be a constant value. While different reward factors may be used here it was found that reward amount based on the NPV did not provide any improvements over this constant reward on a subset of the instances. Hence, for the sake of simplicity, a constant reward was used. ρ is the learning rate as specified earlier and is defined to be 0.1 for this study.

ComputeConvergence(π^{ib}): As a convergence measure we use the iteration best solution and a history of the past θ iteration best solutions to determine if the pheromones have converged. If sampling the pheromones repeatedly produces the same solution, the pheromones are considered to have converged.

A list of the past θ solutions, $l_{\pi^{ib}}$, in the form of a queue is maintained. Every time a new iteration best is generated it is appended to the end of this list while the first solution in the list is removed. The quality of the current iteration best π^{ib} is compared to the quality of all the solutions in the list. If they all have the same objective value the pheromones are re-initialized: $f(\pi^{ib}) = f(k), k \in l_{\pi^{ib}} \Rightarrow \tau_{ij} = 1/n \ \forall i, j$. The list is also re-initialized where all previous solutions are removed.

4.1 Scheduling Schemes

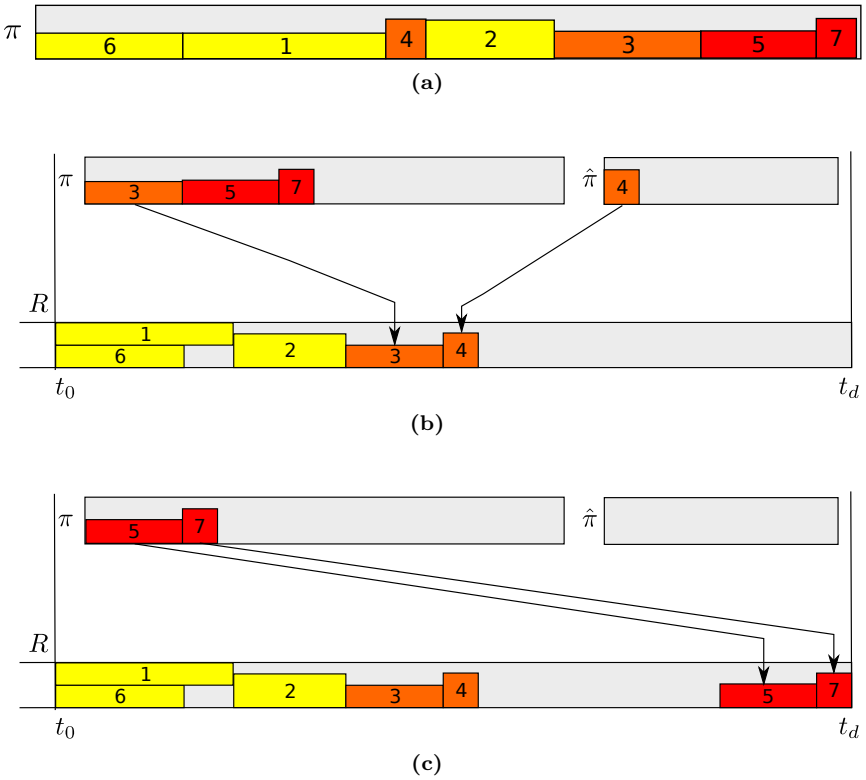


Fig. 1 This figure demonstrates how a permutation of tasks (without precedences) may be mapped to a schedule. There are 7 tasks requiring a single resource and task 3 precedes task 4. Additionally, tasks 5 and 7 have negative-valued cash flows. There are R units of resource available and the height of a task is the amount of resource needed. The time horizon starts at t_0 and finishes at t_d which is the deadline for the tasks. (a) This is a permutation of the 7 tasks. Note that although task 3 must start before task 4, task 4 is allowed to appear ahead of task 3 in the permutation. (b) The positive-valued cash flow tasks are placed as early as possible given the available resource. (c) The negative valued cash flow tasks are placed starting at the end of the schedule. Tasks are successively chosen from the end of the permutation.

Given a permutation, the tasks are scheduled using a serial scheduling scheme. The scheme is similar to the one used by Li and Willis (1992) and a similar modified scheme by Kimms (2001). The permutation is split into two sets, $N = N^+ \cup N^-$. N^+ consists of all tasks with positive cash flows which are independent of other tasks or those tasks whose cash flow is positive and greater than the cumulative cash flow of its dependent tasks:

$$N^+ = \left\{ t : cf(t) - \sum_{t' \in S(t)} cf(t') > 0 \right\} \tag{28}$$

where $S(t)$ is the set of direct and indirect successors of t. N^- is the remaining set of tasks, i.e. $N \setminus N^+$.

We consider two ways of scheduling tasks given a permutation (see Figure 1). Consider the permutation π in Figure 1 (a) where the problem consists of 7 tasks with precedences between two of them (task 3 must finish before task 4 commences). N^- consists of tasks 5 and 7 have negative-valued cash flows. There is a single resource R and the heights of the tasks specify the amount of resource needed. As the permutation shows, precedences are not maintained when constructing the permutation. Figure 1 (b) shows the placement scheme. Here, positive-valued tasks are placed as early as possible, satisfying the resource constraints. If a task appears which has a predecessor that is not scheduled, it is placed on a waiting list $\hat{\pi}$. Once the predecessor is placed (task 3) the task on the waiting list is also immediately placed.

Figure 1 (c) demonstrates how negative-valued cash flow tasks are scheduled. Here, tasks are considered from the end of the permutation and starting at the deadline, tasks are placed in a greedy fashion similar to how the positive-valued cash flows are placed at the beginning of the schedule. Note that the negative valued cash flow tasks do not have to appear at the end of the sequence. If they happen to be in-between positive-valued tasks they are still placed starting at the deadline.

Precedences are not considered when constructing permutations in the above scheme, however, this is trivially accounted for at the scheduling stage. We have attempted to ensure precedences are satisfied within the permutation but found improved results when we ignore them in the permutations.

4.2 LR-ACO

ACO can be incorporated in a straightforward manner into Algorithm 1. In Algorithm 1, ImproveLB(π) can be replaced with Algorithm 3. The basic idea is to seed ACO with π to improve the lower bound. π is used as the global best solution for the ACO procedure which biases the search towards this solution. Note that there are no changes with respect to the manner in which schedules are generated. Lagrangian relaxation with or without ACO uses the same scheme described in the previous section. There are two variants of LR-ACO that we consider. LR-Kimms-ACO is the LR model originally suggested

by Kimms (2001) with ACO and LR-SE-ACO is the more efficient LR model also combined with ACO.

5 Experiments and Results

5.1 Algorithm Settings

Experiments were conducted with LR, ACO, LR-ACO where the LR components make use of the model of Kimms 3.1. The following parameter settings were chosen by conducting tests on a subset of the instances. To choose the number of solutions per iteration, n_a, $\{5, 10, 20, 30\}$ solutions were tested and it was found that 10 was the most effective. Similarly, $q_0 = 0.9$ was determined from $\{0.3, 0.5, 0.7, 0.9, 1.0\}$. This amounts to high deterministic selection. $\rho = 0.1$ was selected from $\{0.1, 0.01\}$ and while this is a relatively high learning rate it is justified given that the pheromones are re-initialized when ACO converges to a single solution. Note that the same settings were used for ACO or all algorithms using an ACO component. In the case where LR is combined with ACO, we have allowed the ACO search 500 iterations. This was not determined in any systematic way but rather selected based on conducted as few "meaningful" iterations as possible. Thus 500 iterations provides a reasonable number of updates to the pheromones in a relatively short time-frame.

The LR algorithm of Kimms (2001) is deterministic and was run once for every instance. All the runs were given at most 15 minutes of execution time. The thresholds were set to gap < 0.01 or $\lambda < 0.01$ below any of which the algorithm will terminate. The experiments were conducted on the Monash Sun Grid and the Enterprisegrid with Nimrod/G (Abramson et al, 2000).

5.2 Benchmark Sets

The first set of instances were obtained from the project scheduling problem library (Kolisch and Sprecher, 1997) which were also the instances used by Kimms (2001). These include a large number of instances with varying degrees of network complexities, resource factors and resource strengths. We first conduct a number of experiments on all the problems instances with 120 tasks to confirm that similar results are being achieved to Kimms (2001) and also to show that LR-Kimms-ACO provides improved results compared to LR and ACO independently. Additionally, we conduct a number of experiments with a subset of the instances (60 and 120 tasks) which are detailed in Section 6.

These instances are categorized in terms of three measures. Firstly, network complexities indicate the proportion of precedences incorporated in the instance with a larger value implying a larger number of precedences. The second measure is the resource factor which specifies how many resources are required by an activity in proportion to the total number of resources available. Finally, the resource strengths measure scarceness of resources with low values implying that resource constraints are tight.

The deadlines δ were determined in a similar manner to Kimms (2001) so that feasible solutions are found easily. Note that in this study we do not aim to minimize makespan so the deadlines chosen here are larger than the previous study. For a task j, a latest start time (ls_j) is determined by recursively considering all preceding tasks, i.e., $ls_j \geq ls_i + d_i \ \forall (i,j) \in P$. Then, $\delta = 3.5 \times max_j ls_j$. The cash flows are determined exactly like Kimms (2001) by selecting $c_i = [-500, 1000]$ uniformly randomly. We examine two discount rates α. The first is determined like Kimms (2001) where $\alpha = \sqrt[52]{1 + 0.05} - 1$ and the second is $\alpha = 0.01$.

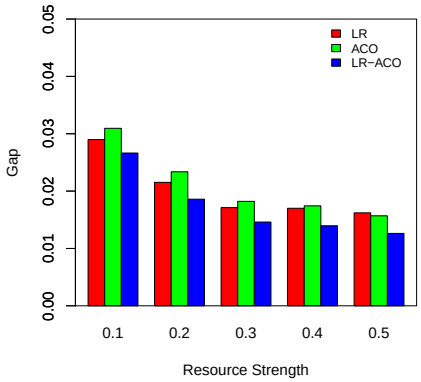
Vanhoucke (2010) also had a number of project scheduling instances with similar characteristics as those tested by Kimms (2001). We consider all the instances with 100 tasks. The major difference between these two studies is that Vanhoucke (2010) uses tight deadlines whereas the schedules generated by Kimms (2001) are always deadline feasible. We use similar settings as those used by Vanhoucke (2010) in terms of deadlines and these experiments are discussed in more detail in Section 5.4.

5.3 Results for Kimms' Instances with 120 Tasks

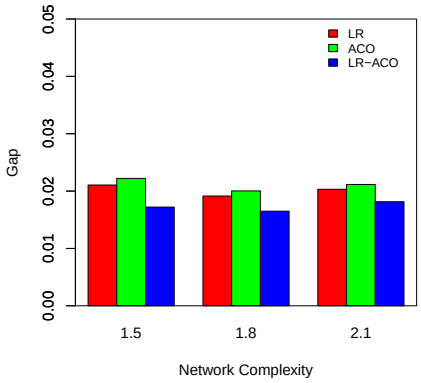
We consider the datasets from the project scheduling problem library with all instances consisting of 120 tasks. Figure 2(a) shows the average results across all instances with 120 tasks. The gap is defined as $(ub - lb)/ub$. For ACO, we have used the LR upper bound to determine its gap. We see that the average gap of 2% for LR is similar to what was seen by Kimms (2001). The average gap obtained by LR-Kimms-ACO is significantly lower than LR and ACO and proves to be the best option. ACO is effective, but repairing solutions provided by LR are slightly more effective.

We now examine the results by resource strength (RS), resource factor (RF) and network complexity (NC). See Figure 2 (b),(c) and (d). The first observation is that LR-Kimms-ACO is always the best performing algorithm always providing average lower gaps than any of the two methods on their own. For the more tightly constrained problems, LR is superior to ACO (large resource factors and low resource strength). This is expected since LR is designed to deal with constraints effectively, whereas ACO and meta-heuristics in general often struggle to deal with hard constraints (Meyer and Ernst, 2004). The resource factor shows that as far as network complexity is concerned, ACO is always marginally worse than LR which, in turn, is worse than LR-Kimms-ACO.

Table 1 shows the breakdown of the results discussed in Figure 2. The table shows lower and upper bounds for each algorithm broken down by resource factors, network complexity and resource strengths. The row $Mean$ shows the average across all instances. The following rows show the averages for each measure. The LR-based algorithms are always superior to ACO concerning the lower bounds. Here, LR-ACO is generally superior to LR whereas LR is more effective on 4 out of the 12 measures. Interestingly, ACO assists to



(a) **(b)**



(c) **(d)**

Fig. 2 The average results of each algorithm for all the problem instances with 120 tasks split by (a) resource strength, (b) resource factor and (c) network complexity

improve the upper bounds quite significantly by outperforming LR across all measures.

Table 1 Breakdown of the results on all the instances with 120 tasks. Standard deviations are provided within parentheses ().

		Lower bounds			Upper bounds	
		ACO	LR	LR-ACO	LR	LR-ACO
Mean		28341.97(4739.13)	28368.64(4726.61)	28373.17(4735.29)	28927.62(4677.53)	28853.41(4700.20)
RS	0.1	27951.99(4975.57)	28004.43(4957.06)	28015.24(4974.52)	28814.63(4951.93)	28760.25(4973.95)
	0.2	27798.58(4618.94)	27848.74(4613.47)	27844.12(4610.27)	28440.15(4580.27)	28355.68(4599.71)
	0.3	28413.23(4891.34)	28443.19(4884.65)	28437.62(4890.00)	28921.53(4863.74)	28846.01(4881.74)
	0.4	28538.21(4443.84)	28549.61(4439.14)	28552.42(4444.94)	29025.22(4396.03)	28942.20(4415.48)
	0.5	29174.54(4673.90)	29157.53(4657.95)	29181.48(4674.47)	29603.96(4542.67)	29526.37(4578.31)
NC	1.5	28303.28(4670.71)	28333.75(4655.13)	28346.20(4665.94)	28910.18(4565.04)	28815.83(4595.05)
	1.8	28090.73(4858.94)	28114.20(4850.50)	28117.54(4860.03)	28641.41(4817.76)	28571.62(4836.01)
	2.1	28731.91(4670.39)	28754.14(4655.76)	28754.79(4663.31)	29331.70(4624.56)	29270.85(4643.82)
RF	0.25	28785.68(5192.35)	28757.21(5182.03)	28791.57(5193.90)	29250.12(5123.28)	29170.47(5146.11)
	0.5	28468.52(4684.87)	28476.65(4677.24)	28496.99(4685.44)	29059.12(4594.80)	28979.27(4623.26)
	0.75	27546.45(4612.40)	27592.71(4600.43)	27586.53(4605.14)	28201.23(4578.29)	28124.12(4602.00)
	1	28700.57(4350.58)	28776.23(4332.60)	28749.61(4342.39)	29333.91(4310.99)	29270.55(4326.17)

5.4 Results for Vanhoucke's Instances with 100 Tasks

For this experiment, we consider the instances from Vanhoucke (2010) who also used tight deadlines. We consider the largest instances of 100 tasks or more. Gu et al (2013) examine a Lagrangian relaxation and constraint programming (CP-LR) hybrid for the same set of instances, however they do not extend deadline to allow feasible solutions. Similar to our results, the CP-LR provides feasibility on more than 99% of the instances with 100 tasks. More interestingly however, the CP-LR algorithm focuses on project scheduling whereas the proposed LR-ACO is applicable to domains beyond scheduling.[6]

Firstly, we examine deadlines of 20% beyond the minimum project deadline. The results are presented in Table 2 and Figure 3 for instances with all tasks having positive cash flows (100-0), 20%, 40%, 60%, 80% negative cash flows to 100% negative cash flows. The figure shows the gaps whereas the table shows a breakdown into lower and upper bounds. Since the deadlines are hard, feasible solutions may not always be found and in this case we extend the deadlines to allow feasibility. Note that this was required for less that 1% of the instances.

We see that for mainly positive cash flows (0% and 20% negative cash flows), LR-ACO is the best algorithm. This is mainly sure to the upper bounds being very tight. For these instances, LR is most effective in finding lower bounds. Beyond 40% negative cash flows, the scatter search is the best lower bounding method. While the tight deadlines can be attributed for this effect, we looked more closely at the individual runs of these methods. We find that even the relaxed problems of the LR require significant solving time which increase with negative cash flow tasks. Thus, in the presence of very tight deadlines, much larger run-times are needed to provide a reasonable number of iterations for the algorithm to converge. Hence, the scatter search is more effective since it is able to generate a large number of solutions (lower bounds) in reduced time-frames.

The conclusion above shows that when there are a large number of negative cash flows ($\geq 40\%$) and tight deadlines, the scatter search is a more effective algorithm. Hence, we investigated tighter deadlines to see if the scatter search will eventually be effective on the 0% and 20% instances. These results are also present in Table 2. We see that with increasing tightness of deadlines, the scatter search is more effective even for the positive cash flow instances.

[6] Both CP and ACO have the potential to improve solutions from LR, however, ACO is more straight-forward to customize and implement.

Table 2 Results on the instances of Vanhoucke (2010) with 100 tasks. The results reported as averages with standard deviations in parentheses () and were obtained after eliminating infeasible runs.

	Scatter search		ACO	LR		LR-ACO	
Instance	LB	UB	LB	LB	UB	LB	UB
Deadline - 20%							
100-00	10901.07(3843.79)	19455.22(2164.97)	10863.90(6071.64)	10980.32(5764.34)	15897.31(7054.68)	10880.68(5837.80)	14388.67(6762.12)
100-20	7004.64(2272.29)	12173.32(1736.62)	7017.24(3821.04)	7023.79(3579.96)	10558.53(4662.58)	6864.16(3560.98)	9507.82(4394.06)
100-40	2901.99(1058.02)	4816.96(1433.44)	2380.09(1441.72)	2608.70(1361.68)	4756.02(2224.58)	2145.62(1326.15)	4387.56(2083.89)
100-60	1211.06(1015.19)	2111.80(1238.88)	272.93(845.66)	647.52(916.44)	2026.65(1450.64)	-81.76(884.15)	1965.37(1390.97)
100-80	-1181.89(2076.65)	97.10(2019.56)	-2963.39(2283.52)	-2629.83(2337.19)	-581.16(2113.56)	-3213.77(2484.16)	-646.83(2030.26)
100-100	-6237.21(4287.57)	-3545.93(3951.75)	-9581.04(5768.97)	-9742.35(5674.34)	-4986.03(4694.44)	-9808.21(5795.04)	-5059.72(4683.22)
Deadline - 15%							
100-00	10893.15(3585.81)	19455.22(2468.44)	10312.79(3554.46)	10396.76(3564.97)	13101.95(2726.87)	10169.95(3560.07)	14394.21(2227.46)
100-20	6978.03(2421.74)	12167.44(3350.60)	6618.22(2361.66)	6618.26(2440.91)	8648.74(2572.13)	6438.68(2338.36)	9441.16(2637.41)
Deadline - 10%							
100-00	10829.61(3568.28)	19455.22(2471.08)	10200.47(3582.09)	10337.48(3582.65)	13152.74(2721.65)	10072.43(3613.28)	14604.69(2171.84)
100-20	6881.57(2373.87)	12161.20(3348.10)	6489.37(2395.78)	6526.37(2403.15)	8673.39(2571.58)	6316.35(2365.31)	9584.47(2646.25)
Deadline - 5%							
100-00	10284.49(4407.38)	19455.22(2474.02)	10160.84(3619.41)	10191.91(3612.31)	14939.87(2085.65)	10030.21(3910.79)	13262.09(3118.14)
100-20	6315.92(3079.64)	12154.34(3345.38)	6415.14(2424.39)	6437.24(2445.22)	9826.33(2664.62)	10021.42(4035.63)	13246.67(4136.61)

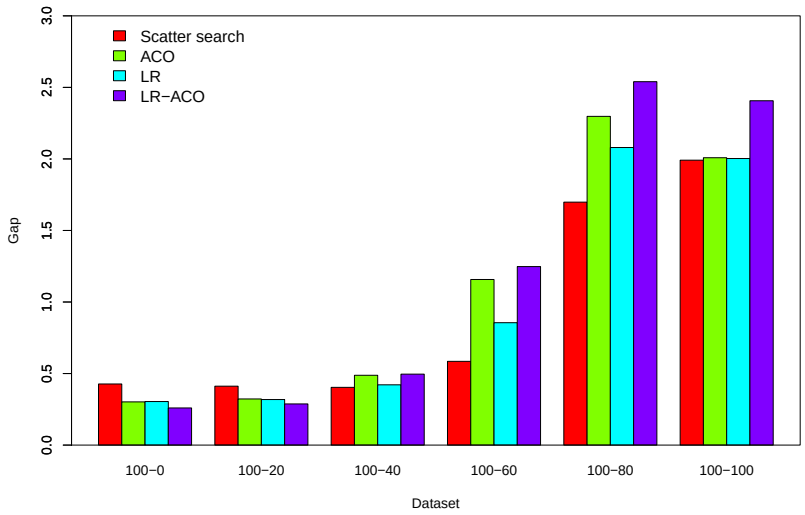


Fig. 3 A comparison of Scatter search Vanhoucke (2010), ACO, LR-ACO and LR+ACO. The x-axis represents the proportion of negative cash flow tasks. 100-0 consists of only positive cash flow tasks whereas 100-100 consist of only negative cash flow tasks.

6 Investigating Algorithms

From among the problem instances of Kimms (2001), we selected 12 instances with 60 and 120 activities with a total of 24 instances.[7] Our aim is to measure the performance of the algorithms on tightly constrained problems and the instances we have chosen reflect this. Thirty runs per instance were conducted for ACO and LR-ACO and each algorithm was given 15 minutes of execution time.

All of the selected instances have a network complexity value of 2.1. Resource factors were chosen from 0.5 and 1.00 where, in the first case, the tasks require half the number of resources available. In the second case the tasks require all the resources available. Resource strengths with values 0.2 or 0.5 were chosen reflect a wide range of resource strengths.

We present the results of the algorithms on the 24 selected instances (see Table 3). There are four categories of instances, each category consisting of 3 instances, with 60 and 120 tasks each. The table specifies network complexities (NC), resource factors (RF) and resource strengths (RS). The tables to follow highlight statistically significant results ($p = 0.05$) with italics and boldface.

[7] We only chose to select a subset of the instances since ACO is stochastic and hence requires several runs on the same instance in order to obtain statistically valid results.

Table 3 Instances selected for comparing the algorithms.

	NC	RF	RS	Tasks
37	2.1	0.5	0.2	60
38	2.1	0.5	0.5	60
45	2.1	1.0	0.2	60
46	2.1	1.0	0.5	60
47	2.1	0.5	0.2	120
50	2.1	0.5	0.5	120
57	2.1	1.0	0.2	120
60	2.1	1.0	0.5	120

Additionally, the best result achieved on any instance is marked only in boldface. The best, mean and standard deviations (sd) for lower bounds (lb) and upper bounds (ub) including gaps where applicable are also reported. In the case of ACO, the gap is reported to the upper bound obtained by LR-SE-ACO.

6.1 Comparing LR-SE-ACO and LR-Kimms-ACO

The first set of results are presented in Table 4 with the aim of determining which of the two models (LR-SE-ACO or LR-Kimms-ACO) is more effective. This table clearly shows that for several instances LR-SE-ACO is more effective considering the lower bound. This is mainly due to the LR-SE model being a stronger formulation, thus solving the relaxed problem more quickly leading to improved results. Thus LR-SE-ACO proves to be the superior model and we therefore make use of this model for further comparisons.

Table 4 The results of LR-SE-ACO and LR-Kimms-ACO. The best, mean and standard deviations for the lower bound (lb) and upper bound (ub) for each algorithm reported. The gap is determined as $(ub - lb)/ub$. Statistically significant results at $p = 0.05$ are italicized and marked in boldface. The best results obtained for any instance are marked in boldface.

	LR-SE-ACO							LR-Kimms-ACO						
	lb-best	lb-mean	sd	ub-best	ub-mean	sd	gap	lb-best	lb-mean	sd	ub-best	ub-mean	sd	gap
37-3	18663.30	***18656.52***	6.63	**18959.50**	***18962.11***	6.63	0.02	18663.30	18646.16	9.92	18985.80	18991.97	9.92	0.02
37-5	14996.50	***14995.30***	0.94	**15203.10**	***15203.85***	0.94	0.01	14996.50	14994.22	1.53	15204.30	15204.95	1.53	0.01
37-8	12138.60	***12135.45***	1.79	**12303.30**	12304.76	1.79	0.01	12138.60	12134.21	1.80	**12304.50**	12305.01	1.80	0.01
38-3	11575.50	11575.50	0.00	11882.20	11882.30	0.00	0.03	11575.50	11575.50	0.00	**11881.60**	11881.89	0.00	0.03
38-5	**7607.46**	7607.35	0.04	7686.89	7686.89	0.04	0.01	7607.34	7607.34	0.00	7686.36	***7686.59***	0.00	0.01
38-8	**17972.20**	17965.32	5.19	18119.40	18119.40	5.19	0.01	17972.00	17962.16	7.08	18119.40	18119.40	7.08	0.01
45-3	13058.80	13051.80	2.77	**13489.00**	***13489.63***	2.77	0.03	**13060.80**	13050.51	4.54	13490.80	13492.52	4.54	0.03
45-5	15888.10	***15885.36***	1.80	**16093.30**	***16095.01***	1.80	0.01	**15888.30**	15883.18	4.23	16097.70	16100.46	4.23	0.01
45-8	**13128.00**	***13118.80***	3.86	**13254.80**	***13256.31***	3.86	0.01	13124.10	13112.11	6.55	13263.90	13271.04	6.55	0.01
46-3	**16614.90**	16606.03	7.39	16667.40	16667.40	7.39	0.00	16614.10	16604.59	8.48	16667.40	16667.40	8.48	0.00
46-5	**15971.60**	***15970.91***	0.92	**16252.90**	***16253.56***	0.92	0.02	15971.50	15969.24	1.57	16255.50	16256.13	1.57	0.02
46-8	**13176.70**	***13175.18***	0.69	13402.00	13402.27	0.69	0.02	13176.10	13174.51	1.01	**13401.80**	13402.30	1.01	0.02
47-3	**28302.00**	***28276.53***	12.42	28977.40	***28978.94***	12.42	0.02	28223.50	28187.23	17.71	29031.10	29049.22	17.71	0.03
47-5	**36059.10**	***36019.83***	17.37	36607.70	***36614.20***	17.37	0.02	36018.20	35973.26	24.12	36726.00	36744.61	24.12	0.02
47-8	**25294.10**	***25250.33***	15.19	25865.40	***25870.71***	15.19	0.02	25237.60	25179.97	22.48	26084.20	26096.96	22.48	0.04
50-3	**28583.00**	***28577.66***	2.71	29109.30	***29110.22***	2.71	0.02	28578.50	28571.33	3.90	29112.30	29112.98	3.90	0.02
50-5	26630.50	26603.72	9.92	26800.70	26800.70	9.92	0.01	**26641.30**	26606.17	12.61	26800.70	26800.70	12.61	0.01
50-8	**30242.10**	***30235.03***	2.99	30647.80	***30648.25***	2.99	0.01	30232.00	30219.72	5.88	30654.90	30656.68	5.88	0.01
57-3	**29835.60**	***29786.93***	15.85	30549.70	***30560.40***	15.85	0.03	29773.20	29716.14	34.63	30920.50	30992.64	34.63	0.04
57-5	**26886.70**	***26840.91***	28.06	27411.80	***27416.08***	28.06	0.02	26753.80	26663.62	34.73	27774.70	27820.41	34.73	0.04
57-8	**26627.80**	***26601.45***	12.67	27108.60	***27110.54***	12.67	0.02	26526.40	26451.24	20.66	27272.10	27358.73	20.66	0.03
60-3	38132.80	38089.72	21.55	38439.60	38447.00	21.55	0.01	**38139.10**	38086.12	19.10	**38438.30**	38447.03	19.10	0.01
60-5	**24988.50**	***24970.89***	6.57	25211.50	***25218.77***	6.57	0.01	24943.30	24926.82	7.68	25234.00	25237.71	7.68	0.01
60-8	**28645.40**	***28637.71***	3.71	29029.80	***29030.23***	3.71	0.01	28619.90	28611.31	3.76	29039.10	29040.46	3.76	0.01
Mean							0.02							0.02

6.2 Comparing ACO, LR-SE and LR-SE-ACO

The second comparison is between ACO, LR-SE and LR-SE-ACO. The results are shown in Table 5. Considering the lower bounds, we see that LR-SE-ACO is the best performing method across most instances, especially for the instances with 60 tasks. ACO is also effective and in some cases obtains the best results (e.g. for instance 50 - 5). LR-SE is generally worse, however, by small margins. For the instances with 120 tasks, a select number of instances in categories 57 and 60 shows that LR-SE is as good or better than the other algorithms on average. However, the best results of LR-SE-ACO and ACO are still superior, with LR-SE-ACO performing best. Additionally, there are no significant differences with the upper bounds and gaps obtained are also of similar levels.

We explain LR-SE-ACO's improved performance by the following. The LR algorithm obtains optimal start times for the tasks given the relaxed problem. The relaxed problem approaches the original problem overtime through the penalties, leading to start times for the tasks close to that of the optimal. ACO is able to use these start times to further explore surrounding regions of the search space effectively through the use of high learning rates. This leads to improvements on most occasions. Where there are no improvements or worse performance by LR-SE-ACO compared to LR-SE, the ACO component has not effective. Here the time spent on the ACO component has not been as useful but where LR iterations may have been.

Table 6 show results for two other variants of LR-SE with ACO. The first one is where the LR solution information is used to bias the ACO selection via the heuristic information (η) referred to as LR-ACO (heur). The second scheme is one where ACO is run with a partially converged pheromone matrix after LR has completed (LR+ACO). These results are presented in Table 6. LR-ACO (heur) is almost always more effective. Comparing with LR-ACO, LR-ACO (heur) performs worse on the small problems while it is slightly more effective on larger instances.

Table 5 The results of LR-SE-ACO, ACO and LR-SE. The best, mean and standard deviations for the lower bound (lb) and upper bound (ub, where applicable) for each algorithm reported. The gap is determined as $(ub - lb)/ub$. Statistically significant results at $p = 0.05$ are italicized and marked in boldface. The best results obtained for any instance are marked in boldface.

	LR-SE-ACO							ACO				LR-SE		
	lb-best	lb-mean	sd	ub-best	ub-mean	sd	gap	lb-best	lb-mean	sd	gap	lb	ub	gap
37-3	**18663.30**	***18656.52***	6.63	18959.50	18962.11	6.63	0.02	18663.20	18637.41	11.84	0.02	18516.60	18960.50	0.02
37-5	**14996.50**	***14995.30***	0.94	15203.10	15203.85	0.94	0.01	14996.50	14992.02	3.54	0.01	14958.70	15203.30	0.02
37-8	**12138.60**	***12135.45***	1.79	12303.30	12304.76	1.79	0.01	12133.70	12124.80	5.32	0.01	12115.00	12304.50	0.02
38-3	**11575.50**	11575.50	0.00	11882.20	11882.30	0.00	0.03	11575.50	11574.05	4.07	0.03	11545.70	11882.30	0.03
38-5	**7607.46**	***7607.35***	0.04	7686.89	7686.89	0.04	0.01	7607.46	7607.16	0.16	0.01	7593.65	7686.88	0.01
38-8	17972.20	17965.32	5.19	18119.40	18119.40	5.19	0.01	**17974.70**	***17972.27***	3.03	0.01	17950.40	18114.70	0.01
45-3	13058.80	***13051.80***	2.77	13489.00	13489.63	2.77	0.03	**13060.60**	13046.46	6.74	0.03	13003.20	13490.30	0.04
45-5	15888.10	***15885.36***	1.80	16093.30	16095.01	1.80	0.01	**15889.80**	15872.01	16.85	0.01	15797.40	16093.90	0.02
45-8	**13128.00**	***13118.80***	3.86	13254.80	13256.31	3.86	0.01	13127.60	13107.06	13.40	0.01	13070.80	13253.90	0.01
46-3	16614.90	16606.03	7.39	16667.40	16667.40	7.39	0.00	**16617.30**	***16611.88***	5.18	0.00	16548.50	16667.40	0.01
46-5	**15971.60**	***15970.91***	0.92	16252.90	16253.56	0.92	0.02	15971.60	15960.94	8.19	0.02	15938.40	16253.10	0.02
46-8	**13176.70**	***13175.18***	0.69	13402.00	13402.27	0.69	0.02	13175.30	13168.60	4.20	0.02	13155.90	13402.40	0.02
47-3	**28302.00**	***28276.53***	12.42	28977.40	28978.94	12.42	0.02	28236.60	28186.93	28.56	0.03	28271.50	28978.40	0.02
47-5	36059.10	36019.83	17.37	36607.70	36614.20	17.37	0.02	**36065.00**	36010.52	34.40	0.02	35922.90	36615.10	0.02
47-8	**25294.10**	25250.33	15.19	25865.40	25870.71	15.19	0.02	25293.30	25240.15	29.59	0.02	25247.20	25867.80	0.02
50-3	28583.00	***28577.66***	2.71	29109.30	29110.22	2.71	0.02	**28588.90**	28568.01	10.91	0.02	28572.80	29109.50	0.02
50-5	26630.50	26603.72	9.92	26800.70	26800.70	9.92	0.01	**26687.20**	***26655.47***	17.95	0.01	26593.90	26800.70	0.01
50-8	30242.10	30235.03	2.99	30647.80	30648.25	2.99	0.01	**30244.60**	30226.86	9.78	0.01	30218.60	30647.90	0.01
57-3	29835.60	29786.93	15.85	30549.70	30560.40	15.85	0.03	**29858.70**	29774.50	45.42	0.03	29768.50	30565.50	0.03
57-5	**26886.70**	26840.91	28.06	27411.80	27416.08	28.06	0.02	26871.10	26781.14	52.48	0.02	26853.90	27416.40	0.02
57-8	**26627.80**	26601.45	12.67	27108.60	27110.54	12.67	0.02	26568.30	26493.92	35.18	0.02	26614.60	27112.20	0.02
60-3	38132.80	38089.72	21.55	38439.60	38447.00	21.55	0.01	**38161.40**	***38128.96***	19.30	0.01	38052.30	38430.10	0.01
60-5	**24988.50**	24970.89	6.57	25211.50	25218.77	6.57	0.01	24981.00	24950.58	14.34	0.01	24972.30	25216.80	0.01
60-8	28645.40	28637.71	3.71	29029.80	29030.23	3.71	0.01	28631.90	28610.05	13.04	0.01	28641.60	29030.50	0.01
Mean							0.02				0.02			0.02

Table 6 LR-SE-ACO (heur) and LR-SE+ACO. The best, mean and standard deviations for the lower bound (lb) and upper bound (ub) for each algorithm reported. The gap is determined as $(ub - lb)/ub$. Statistically significant results at $p = 0.05$ for the lower bounds are italicized and marked in boldface. The best results obtained for any instance are marked in boldface.

	LR-ACO (heur)							LR+ACO				
	lb-best	lb-mean	sd	ub-best	ub-mean	sd	gap	lb-best	lb-mean	sd	ub-mean	gap
37-3	**18663.30**	***18646.71***	8.19	18959.10	18962.02	1.76	0.02	18656.20	18632.02	16.31	18960.50	0.02
37-5	**14995.97**	***14991.31***	4.41	15203.10	15203.75	0.37	0.01	14995.20	14989.12	3.44	15203.30	0.01
37-8	**12138.66**	***12129.15***	4.52	12304.00	12304.67	0.40	0.01	12137.40	12126.17	6.52	12304.50	0.01
38-3	11575.57	11575.53	0.02	11882.20	11882.30	0.02	0.03	11575.50	11575.50	0.00	11882.30	0.03
38-5	7607.34	7607.22	0.13	7686.88	7686.89	0.00	0.01	7607.34	7607.15	0.16	7686.88	0.01
38-8	17971.87	17962.24	6.27	18119.40	18119.40	0.00	0.01	**17974.70**	***17972.16***	3.28	18114.70	0.01
45-3	**13067.78**	13040.14	8.72	13489.10	13489.73	0.46	0.03	13062.70	***13047.52***	9.11	13490.30	0.03
45-5	**15889.52**	***15876.71***	13.41	16092.70	16095.09	1.02	0.01	15884.90	15863.09	18.20	16093.90	0.01
45-8	**13124.55**	***13115.25***	7.46	13254.40	13255.96	0.86	0.01	13120.80	13105.65	10.24	13253.90	0.01
46-3	16606.31	16598.41	8.46	16667.40	16667.40	0.00	0.00	**16617.90**	***16613.39***	3.33	16667.40	0.00
46-5	**15971.63**	***15967.61***	5.36	16252.90	16253.67	0.48	0.02	15970.50	15959.28	9.15	16253.10	0.02
46-8	13176.37	13168.79	3.96	13402.00	13402.29	0.15	0.02	13175.60	13167.41	5.96	13402.40	0.02
47-3	28311.74	***28291.00***	12.42	28977.10	28978.93	1.01	0.02	**28312.80**	28246.79	23.67	28978.40	0.03
47-5	**36146.75**	***36076.35***	30.44	36610.80	36615.74	3.94	0.01	36070.30	36019.81	31.06	36615.10	0.02
47-8	25335.46	***25297.65***	18.06	25866.90	25870.48	2.56	0.02	**25382.60**	25260.89	47.10	25867.80	0.02
50-3	**28599.90**	***28589.49***	5.47	29109.30	29110.20	0.48	0.02	28589.70	28572.18	8.48	29109.50	0.02
50-5	26646.45	26613.25	11.22	26800.70	26800.70	0.00	0.01	**26694.60**	***26658.41***	18.53	26800.70	0.01
50-8	**30251.15**	***30240.35***	4.75	30648.10	30648.51	0.30	0.01	30246.30	30228.95	8.65	30647.90	0.01
57-3	**29917.54**	***29844.33***	31.29	30548.10	30563.54	6.83	0.02	29903.70	29799.34	46.00	30565.50	0.03
57-5	**26956.88**	***26898.01***	28.47	27411.30	27416.28	2.71	0.02	26929.60	26816.41	54.15	27416.40	0.02
57-8	**26651.11**	***26622.58***	12.96	27108.50	27110.68	1.24	0.02	26595.50	26528.47	29.34	27112.20	0.02
60-3	38110.35	38076.44	14.94	38422.90	38442.92	7.51	0.01	**38173.80**	***38138.72***	16.49	38430.10	0.01
60-5	**24985.79**	***24972.90***	5.10	25214.50	25222.09	3.73	0.01	24982.30	24949.59	15.24	25216.80	0.01
60-8	**28650.67**	***28645.04***	3.58	29029.60	29030.24	0.33	0.01	28632.30	28612.08	11.02	29030.50	0.01
Mean					0.02					0.02		

Now we compare the algorithms with a discount rate of $\alpha = 0.01$. The results are shown in Table 7. As a result of this new discount rate, the gaps obtained are not as close as before in the previous comparisons. In this situation, the advantage gained by LR-SE-ACO is accentuated, providing the best lower bounds across all instances with 60 tasks. For the problems with 120 tasks LR-SE-ACO is still the best performing method, however, its advantage is reduced. Here, ACO is able to outperform LR-SE-ACO occasionally. It is worth noting that even when this is the case, LR-SE-ACO always performs better on average.

The upper bounds obtained by both algorithms are closely matched with LR-SE having a slight advantage. However, observing the gaps shows us that in all cases LR-SE-ACO is superior except for two instances 50-3 and 60-5. This is easily explained by the advantage LR-SE-ACO obtains from ACO improvement to the lower bounds.

To summarize the results presented above, LR-SE-ACO is the most effective algorithm to obtain lower bounds. Specifically, the LR component provides a very good seed for the ACO component which is quick to improve the solutions found. Clearly either algorithm, ACO or LR-SE, on their own are not as effective, whereas the hybrid method is best suited for this problem.

Table 7 The results of LR-SE-ACO, ACO and LR-SE with $\alpha = 0.01$. The best, mean and standard deviations for the lower bound (lb) and upper bound (ub, where applicable) for each algorithm reported. The gap is determined as $(ub - lb)/ub$. Statistically significant results at $p = 0.05$ are italicized and marked in boldface. The best results obtained for any instance are marked in boldface.

	LR-SE-ACO							ACO				LR-SE		
	lb-best	lb-mean	sd	ub-best	ub-mean	sd	gap	lb-best	lb-mean	sd	gap	lb	ub	gap
37-3	11222.90	***11198.41***	20.70	12688.90	12692.48	20.70	0.12	**11236.00**	11114.76	56.43	0.12	10905.50	12689.30	0.14
37-5	**10157.60**	***10149.71***	3.14	11231.80	11234.13	3.14	0.10	10149.20	10132.05	16.02	0.10	9983.34	11234.40	0.11
37-8	**9100.75**	***9081.60***	11.13	9683.09	9690.76	11.13	0.06	8948.59	8880.20	29.65	0.08	8992.95	9684.32	0.07
38-3	**9018.26**	***9018.26***	0.00	9293.68	9294.35	0.00	0.03	8251.25	8247.52	15.58	0.11	8955.12	***9293.65***	0.04
38-5	**4851.04**	***4851.04***	0.00	5208.56	5208.58	0.00	0.07	4851.04	4848.58	2.58	0.07	4751.08	***5208.58***	0.09
38-8	**13779.30**	***13779.03***	0.64	14348.10	14349.00	0.64	0.04	13779.30	13760.38	23.07	0.04	13712.90	14348.50	0.04
45-3	**10689.30**	***10654.04***	22.74	11692.10	11698.75	22.74	0.09	10451.40	10372.40	39.09	0.11	10406.20	11690.90	0.11
45-5	**12005.50**	***11998.70***	4.68	12651.90	12660.00	4.68	0.05	11754.20	11693.75	77.16	0.08	11407.30	***12647.20***	0.10
45-8	**8758.08**	***8738.97***	7.28	9409.37	9417.30	7.28	0.07	8747.83	8658.62	68.23	0.08	8378.41	9408.58	0.11
46-3	**12840.60**	***12834.62***	4.29	13046.60	13048.63	4.29	0.02	12838.40	12810.09	21.14	0.02	12684.60	13047.30	0.03
46-5	**11573.40**	***11564.73***	7.80	12395.80	12398.44	7.80	0.07	11454.50	11402.72	42.68	0.08	11426.40	12397.90	0.08
46-8	**10356.00**	***10341.00***	6.44	10552.00	10552.87	6.44	0.02	9825.68	9766.61	34.43	0.07	10250.70	10552.90	0.03
47-3	**19720.30**	19548.64	56.64	21354.50	21358.01	56.64	0.08	19092.00	18878.74	137.40	0.12	19510.40	21358.10	0.09
47-5	**21165.50**	***20991.76***	73.04	23752.60	23756.51	73.04	0.12	**21419.90**	20860.28	209.92	0.12	20733.80	23754.80	0.13
47-8	**14251.30**	14133.96	58.67	16568.90	16571.99	58.67	0.15	**14315.90**	13953.90	179.18	0.16	14079.40	16576.80	0.15
50-3	**20085.30**	20045.72	14.86	21002.50	21004.13	14.86	0.05	19688.90	19590.18	66.58	0.07	20040.50	20997.60	0.05
50-5	**17125.20**	17080.07	19.31	17669.70	17670.75	19.31	0.03	**17192.20**	16916.95	116.15	0.04	17066.60	17670.40	0.03
50-8	**21781.40**	***21763.61***	11.18	22752.80	22753.36	11.18	0.04	21729.20	21644.96	48.25	0.05	21673.20	22747.80	0.05
57-3	**13719.80**	***13527.96***	70.33	16055.30	16071.34	70.33	0.16	13647.40	13398.09	192.22	0.17	13152.00	16059.30	0.18
57-5	**14345.30**	***14211.44***	81.82	16249.90	16260.98	81.82	0.13	**14376.70**	13992.62	179.85	0.14	14006.80	16273.30	0.14
57-8	**15435.10**	15247.77	77.37	17258.40	17269.09	77.37	0.12	15093.40	14871.42	152.32	0.14	15271.70	17261.90	0.12
60-3	**26299.40**	26189.07	37.37	26992.30	26997.25	37.37	0.03	26169.50	25993.54	117.81	0.04	26221.60	26970.40	0.03
60-5	**16034.60**	15996.36	20.34	16741.40	16746.07	20.34	0.05	15903.30	15674.06	133.58	0.06	16008.50	16666.90	0.04
60-8	**21263.80**	21242.29	17.52	22079.50	22080.13	17.52	0.04	20933.20	20819.81	80.76	0.06	21246.90	22069.60	0.04
Mean							0.07				0.09			0.08

6.3 Convergence of Lower Bounds

We further analyze the performance of the algorithms concerning lower bounds.[8] We consider the instances with 120 tasks and report results by averaging their performance across the categories and across all instances. As we have done earlier, 30 runs per instance for ACO and LR-SE-ACO have been conducted the results are mean normalized so that we can compare across instances. The first comparison is of the lower bounds obtained by all the algorithms over 15 minutes and seen in Figure 4. Here, the results for every instance and for all 30 runs per instance are first mean normalized. The mean normalized results are averaged and the difference of each algorithm to the average is reported.

Figure 4 shows that LR-SE-ACO overall is the best performing algorithm. ACO and LR-SE-ACO have a significant advantage at the initial stages. LR-SE improves gradually until 100 seconds, however, at this stage it outperforms ACO. The trend shows that ACO and LR-SE-ACO continue to improve over time whereas LR-SE and LR-SE-ACO (heur) reach a point after which its performance stagnates. These results split by category are shown in Figure 5. Here a similar picture as that of the overall average performance can be seen. However, we note that when the resource factor is less (instances 47 and 50) LR-SE-ACO has an advantage early. LR-SE-ACO always has the initial advantage, but with large resource factors (instances 57 and 60) the algorithms are more closely matched until 200 seconds. From this point LR-SE-ACO re-gains its advantage.

The initial difference is attributable to ACO's ability to improve a starting solution effectively. However, overtime ACO's influence is reduced but still assists LR-SE-ACO. This leads to the divergence of LR-SE-ACO and LR-SE curves gradually. ACO's initial result is very good and it gradually improves but not enough to keep pace with the LR-base algorithms.

Interestingly LR-SE-ACO (heur) follows LR-SE despite using heuristic information. For these runs with $\alpha = 0.01$, the heuristic information seems to make no significant difference unlike the results seen in Table 6. In contrast, LR-SE-ACO is always more effective. Thus, depending on the learning rate, providing a bias via a solution (LR-SE-ACO) which is a stronger form of bias, is more effective than a subtle bias through the heuristic information. Larger discount rates require a stronger bias.

The next comparison of interest is to consider iterations as opposed to time. Since a single iteration of LR-SE-ACO is more expensive than that of LR-SE, the aim here is to identify if it converges more quickly compared to LR-SE given the same number of relaxed problems solved. We consider 100 iterations and comparisons made based on mean normalized data (see Figure 6). LR-SE-ACO is overall the best performing algorithm. This is not surprising given the results previously seen, however, we find that both LR algorithms converge to a point (about 50 iterations) after which they diverge

[8] Note that there were not many significant differences with the upper bounds obtained by the LR algorithms and are hence not analyzed here.

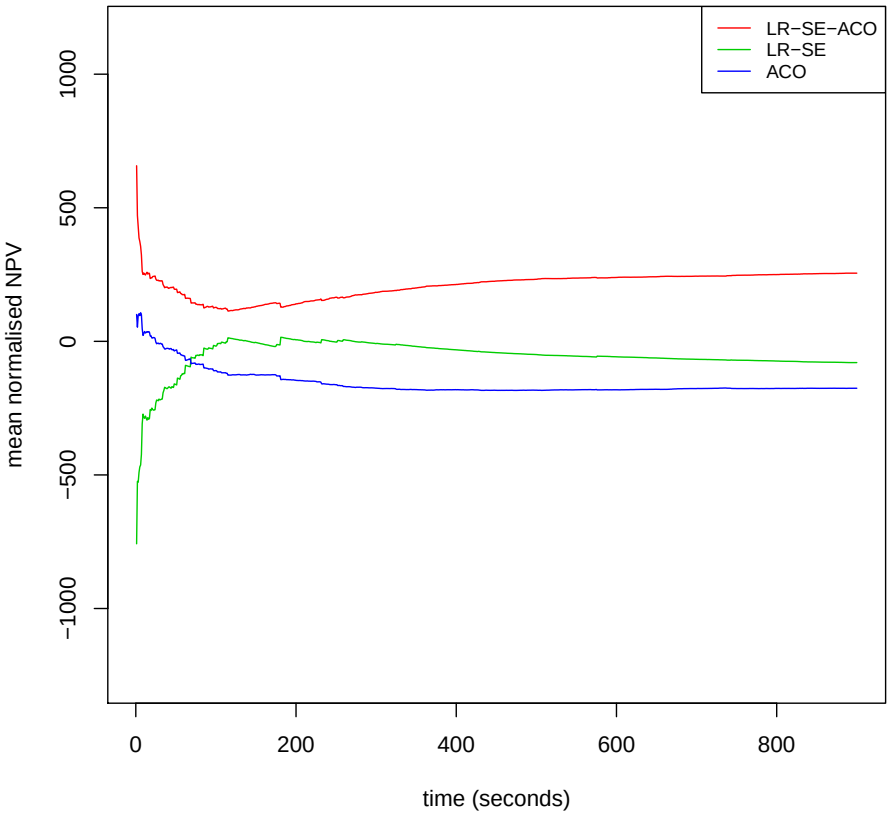
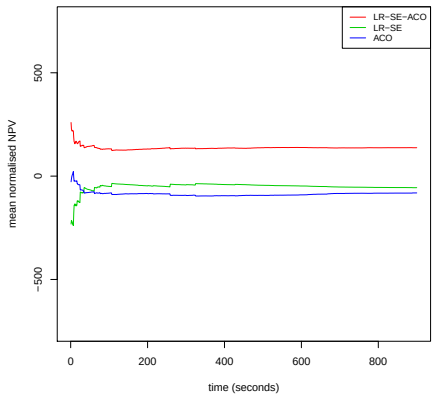


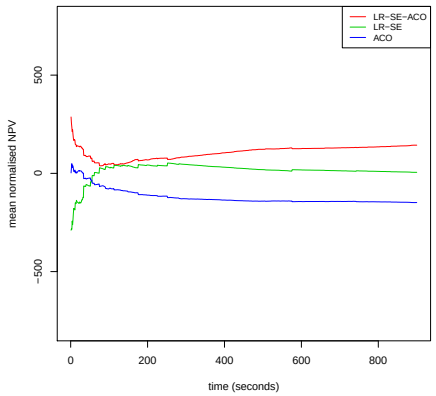
Fig. 4 A comparison of mean normalized lower bounds (NPV) of four algorithms over 15 minutes. The results across the 30 runs are first mean normalized by instance. These results are averaged across all algorithms and the difference to the average is shown for each algorithm.

gradually. This explanation is similar to the one provided earlier where ACO provides large improvements initially but its influence eventually reduces. ACO is the least expensive method per iteration and, not surprisingly, this can be seen across the iterations where ACO is significantly worse. Over 100 iterations we also see that LR-SE-ACO (heur) does not provide significant improvements over LR-SE-ACO which follows a similar trend to what was seen in the long runs.

A break down by category (Figure 7) shows a similar trend. However, category 57 with high resource factor and low resource strength shows that the initial advantage provided by ACO is lost, but gained again across iterations.



47 50



57 60

Fig. 5 A comparison of mean normalized lower bounds (NPV) of four algorithms by category over 15 minutes. Left-right and top-bottom we have the results for instance categories 47, 50, 57, 60. The results across the 30 runs are first mean normalized by instance. These results are then averaged across all algorithms in a category. The difference to the average by category is shown for each algorithm.

7 Conclusion

In this study we have shown that hybrids of a Lagrangian relaxation based heuristic with ACO can be effectively applied to a resource constrained project scheduling problem. We show through comparisons based on CPU time that LR-SE-ACO outperforms LR-SE (Lagrangian relaxation with list scheduling) and ACO on their own when maximizing NPV. The complementary advantages of each algorithm assist the hybrid to improve upon either on their own. This study shows that improvements can be made with respect to lower

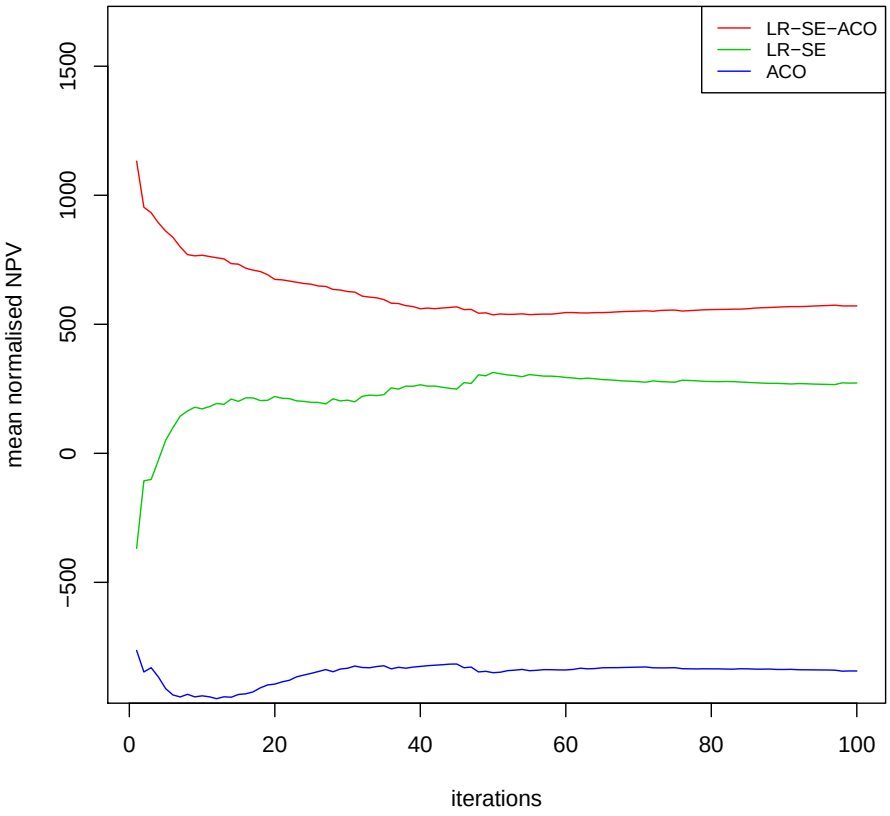
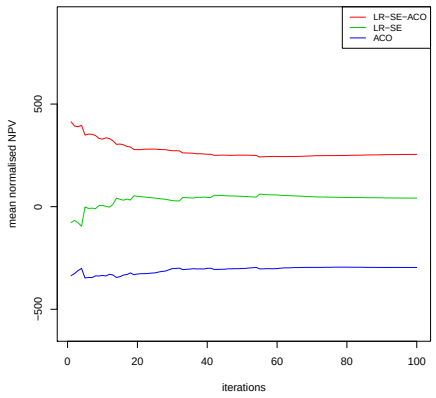


Fig. 6 A comparison of mean normalized lower bounds (NPV) of four algorithms over 100 iterations. The results across the 30 runs are first mean normalized by instance. These results are averaged across all algorithms and the difference to the average is shown for each algorithm.

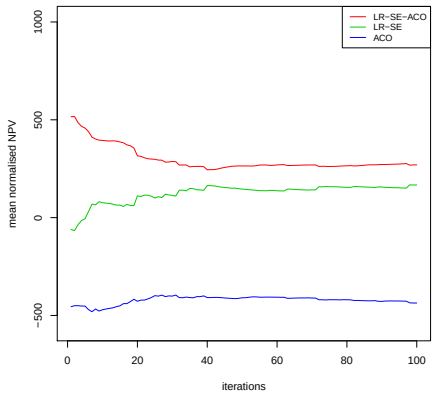
bounds. While the upper bounds may already be very good, improving them still warrants further research and is currently being addressed.

We find that if the project deadlines are tight, the hybrid LR-ACO is not as effective. This is mainly due to the increased solve time of the sub-problems. In this direction, improvements in the original formulation could potentially help to allow the algorithm to execute more quickly in reduced time-frames thereby allowing more iterations and improvements to the lower and upper bounds.

ACO has been shown to assist the LR algorithm, however, other methods may be substituted for ACO effectively. In particular, local search methods may prove effective here to essentially explore the surrounding regions of the



47 50



57 60

Fig. 7 A comparison of mean normalized lower bounds (NPV) of four algorithms by category over 100 iterations. Left-right and top-bottom we have the results for instance categories 47, 50, 57, 60. The results across the 30 runs are first mean normalized by instance. These results are then averaged across all algorithms in a category. The difference to the average by category is shown for each algorithm.

search space suggested by the start times provided by the LR algorithm. Additionally, further feedback from ACO to the LR components can provide more effective bounds and also help the algorithm converge more quickly.

Given these results, extending all of these algorithms to significantly larger problems should provide interesting results. In fact, we are currently exploring data for a problem obtained from the Australian mining industry with up to 11,000 tasks with promising initial results.

Table 8 ACO using default model and ACO using sum rule. The best, mean and standard deviations for the lower bound (lb) for each algorithm reported. Statistically significant results at $p = 0.05$ are italicized and marked in boldface. The best results obtained for any instance are marked in boldface.

	ACO				ACO - sum rule			
	lb-best	lb-mean	sd	gap	lb-best	lb-mean	sd	gap
37-3	11236.00	11114.76	56.43	0.12	11209.70	11116.43	51.05	0.12
37-5	10149.20	10132.05	16.02	0.10	10149.20	10122.62	22.50	0.10
37-8	8948.59	8880.20	29.65	0.08	8918.20	8873.02	39.60	0.08
38-3	8251.25	8247.52	15.58	0.11	8251.25	8242.17	25.69	0.11
38-5	4851.04	4848.58	2.58	0.07	4851.04	4847.78	3.06	0.07
38-8	13779.30	13760.38	23.07	0.04	13779.20	13758.29	26.13	0.04
45-3	10451.40	10372.40	39.09	0.11	10499.50	10370.21	48.60	0.11
45-5	11754.20	11693.75	77.16	0.08	11750.30	11678.64	85.16	0.08
45-8	8747.83	8658.62	68.23	0.08	8747.83	8664.89	62.50	0.08
46-3	12838.40	12810.09	21.14	0.02	12834.50	12810.6	13.69	0.02
46-5	11454.50	11402.72	42.68	0.08	11454.50	11406.23	38.54	0.08
46-8	9825.68	9766.61	34.43	0.07	9814.29	9767.94	33.81	0.07
47-3	19092.00	18878.74	137.40	0.12	19094.80	18850.93	152.27	0.12
47-5	21419.90	20860.28	209.92	0.12	21158.00	20830.91	177.95	0.12
47-8	14315.90	***13953.90***	179.18	0.16	14082.10	13775.73	185.72	0.17
50-3	19688.90	19590.18	66.58	0.07	19735.80	19571.53	73.26	0.07
50-5	17192.20	16916.95	116.15	0.04	17052.40	16873.05	122.04	0.05
50-8	21729.20	***21644.96***	48.25	0.05	21709.00	21602.4	56.39	0.05
57-3	13647.40	13398.09	192.22	0.17	13732.70	13398.8	183.47	0.17
57-5	14376.70	13992.62	179.85	0.14	14239.10	13929.36	190.91	0.14
57-8	15093.40	14871.42	152.32	0.14	15162.00	14797.14	174.33	0.14
60-3	26169.50	***25993.54***	117.81	0.04	26221.70	25919.52	128.21	0.04
60-5	15903.30	15674.06	133.58	0.06	15941.00	15698.72	113.42	0.06
60-8	20933.20	***20819.81***	80.76	0.06	20910.50	20774.98	66.73	0.06
Mean				0.09				0.09

A ACO using Sum Rule

Table 8 shows ACO with the default model compared with ACO with the sum rule. The default model consists of a distribution for each position or variable from which a task is selected. In the sum rule, a task is selected by summing over the pheromone values of that task over the preceding positions:

$$\mathbf{P}(\pi_i = k) = \frac{[\sum_{l=1}^{i} \tau_{lk}]\eta_k^\beta}{\sum_{j \in \mathcal{J} \setminus \{\pi_1, \ldots, \pi_{i-1}\}} \left([\sum_{l=1}^{i} \tau_{lj}]\eta_j^\beta \right)} \tag{29}$$

References

Abramson D, Giddy J, Kotler L (2000) High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In: In International Parallel and Distributed Processing Symposium (IPDPS), IEEE Computer Society, Washington, DC, USA, pp 520–528

den Besten M, Stűtzle T, Dorigo M (2000) Ant Colony Optimization for the Total Weighted Tardiness Problem. Lecture Notes in Computer Science 1917:611–620

Brucker P, Drexl A, Mohring R, Neumann K, Pesch E (1999) Resource-constrained Project Scheduling: Notation, Classification, Models, and Methods. European Journal of Operational Research 112:3–41

Camazine S, Deneubourg JL, Franks NR, Sneyd J, Theraulaz G, Bonabeau E (2001) Self-Organization in Biological Systems. Princeton University Press, Princeton, NJ, USA

Chen WN, Zhang J, Chung HSH, z Huang R, Liu O (2010) Optimizing Discounted Cash Flows in Project Scheduling - An Ant Colony Optimization Approach. IEEE Transactions on Systems, Man, and Cybernetics, Part C 40(1):64–77

Demeulemeester E, Herroelen W (2002) Project Scheduling: A Research Handbook. Kluwer, Boston, MA, USA

Dorigo M (1992) Optimization, Learning and Natural Algorithms. PhD thesis, Dip. Elettronica

Dorigo M, Gambardella LM (1997) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions On Evolutionary Computation 1:53–66

Dorigo M, Stűtzle T (2004) Ant Colony Optimization. MIT Press, Cambridge, Massachusetts, USA

Fisher M (2004) The Lagrangian Relaxation Method for Solving Integer Programming Problems. Management Science 50(12):1861–1871

Gu H, Schutt A, Stuckey P (2013) A lagrangian relaxation based forward-backward improvement heuristic for maximising the net present value of resource-constrained projects. In: Gomes C, Sellmann M (eds) Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, Lecture Notes in Computer Science, vol 7874, Springer Berlin Heidelberg, pp 340–346

Kimms A (2001) Maximizing the Net Present Value of a Project Under Resource Constraints Using a Lagrangian Relaxation Based Heuristic with Tight Upper Bounds. Annals of Operations Research 102:221–236

Kolisch R, Sprecher A (1997) PSPLIB a project scheduling problem library. European Journal of Operational Research 96:205–216

Li KY, Willis RJ (1992) An Iterative Scheduling Technique for Resource-constrained Project Scheduling. European Journal of Operational Research 56(3):370–379

Merkle D, Middendorf M, Schmeck H (2000) Ant Colony Optimization for Resource-Constrained Project Scheduling. IEEE Transactions on Evolutionary Computation 6(4):893–900

Meyer B, Ernst A (2004) Integrating ACO and Constraint Propagation. Lecture Notes in Computer Science: Ant Colony, Optimization and Swarm Intelligence 3172/2004:166–177

Neumann K, Schwindt C, Zimmermann J (2003) Project Scheduling with Time Windows and Scarce Resources. Springer, Berlin, Germany

Show YY (2006) Ant Colony Algorithm for Scheduling Resource Cconstrained Projects with Discounted Cash Flows. In: In Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, IEEE, Dalain, China, pp 176–180

Singh G, Ernst AT (2011) Resource Constraint Scheduling with a Fractional Shared Resource. Operations Research Letters 39(5):363 – 368

Stűtzle T, Hoos HH (2000) MAX-MIN Ant System. Future Generation Computer Systems 16:889–914

Thiruvady D, Singh G, Ernst AT, Meyer B (2012) Constraint-based ACO for a Shared Resource Constrained Scheduling Problem. International Journal of Production Economics 141(1):230–242

Thiruvady D, Ernst AT, Singh G (2014) Parallel Ant Colony Optimization for Resource Constrained Job Scheduling. Annals of Operations Research pp 1–18

Vanhoucke M (2010) A Scatter Search Heuristic for Maximizing the Net Present Value of a Resource-constrained Project with Fixed Activity Cash Flows. International Journal of Production Research 48:1983–2001

Vanhoucke M, Demeulemeester E, Herroelen W (2001) On Maximizing the Net Present Value of a Project under Renewable Resource Constraints. Management Science 47(8):1113–1121

Author/s:
Thiruvady, D;Wallace, M;Gu, H;Schutt, A

Title:
A lagrangian relaxation and ACO hybrid for resource constrained project scheduling with discounted cash flows

Date:
2014-12

Citation:
Thiruvady, D., Wallace, M., Gu, H. & Schutt, A. (2014). A lagrangian relaxation and ACO hybrid for resource constrained project scheduling with discounted cash flows. JOURNAL OF HEURISTICS, 20 (6), pp.643-676. https://doi.org/10.1007/s10732-014-9260-3.

Persistent Link:
http://hdl.handle.net/11343/283310