A Lagrangian Relaxation Based Forward-backward Improvement Heuristic for Maximising the Net Present Value of Resource-Constrained Projects

Hanyu Gu, Andreas Schutt, and Peter J. Stuckey

National ICT Australia, Department of Computing and Information Systems,
The University of Melbourne, Victoria 3010, Australia
{hanyu.gu, andreas.schutt, peter.stuckey}@nicta.com.au

Abstract. In this paper we propose a forward-backward improvement heuristic for the variant of resource-constrained project scheduling problem aiming to maximise the net present value of a project. It relies on the Lagrangian relaxation method to generate an initial set of schedules which are then improved by the iterative forward/backward scheduling technique. It greatly improves the performance of the Lagrangian relaxation based heuristics in the literature and is a strong competitor to the best meta-heuristics. We also embed this heuristic into a state-of-the-art CP solver. Experimentation carried out on a comprehensive set of test data indicates we compare favorably with the state of the art.

1 Introduction

We study the Resource-constrained Project Scheduling Problem with Discounted Cashflow (RcpspDc) denoted as $m,1|cpm,\delta_n,c_j|npv$ by [8] or $PS|prec|\sum C_j^F\beta^{C_j}$ by [1]. Specifically, given a set of activities J with precedence relationship $(i,j) \in L$, $i \in J$, $j \in J$, we need to decide the activity start time s_j , $j \in J$ within the project deadline T so that the net present value (NPV) of the project is maximised while the capacity of each renewable resource R_k, $k \in R$ is not violated. Each activity j has an associated cash-flow c_j and requires r_{jk} unit of resource $k \in R$ for a continuous period of time p_j. The net present value is calculated as the sum of the discounted cash flow of each activity defined as $c_j e^{-\alpha(s_j+p_j)}$ where α is the discount rate. A conceptual model can be formulated as

$$NPV = \text{maximise } \textstyle\sum_{j\in J} c_j e^{-\alpha(s_j+p_j)} \tag{1}$$

$$\text{subject to} \quad s_i + p_i \le s_j \qquad \forall (i,j) \in L \tag{2}$$

$$\textstyle\sum_{j\in S(t)} r_{jk} \le R_k \qquad k \in R, t = 0,\cdots,T-1 \tag{3}$$

$$0 \le s_j \le T - p_j \qquad j \in J \tag{4}$$

where $S(t)$ is the set of activities running in period $[t, t+1)$.

The RcpspDc belongs to the class of NP-hard problems, and has been intensively investigated since it was first introduced in [17]. The reader is referred to [7]

for an extensive literature overview of solution approaches for RcpspDc. Significant progress has been made in recent years for both complete and incomplete methods. The lazy clause generation approach to RcpspDc [19] provides the state of the art complete method and outperforms the traditional branch-and-bound based methods [9,24,15]. For larger problems the evolutionary population based scatter search algorithm [23] achieved the bests results in comparison with other meta-heuristics such as genetic algorithms [11] and tabu search [25].

In spite of our success on the application of the Lagrangian Relaxation based Heuristic (LRH) [4] for very large problems (1400-10000 activities) and its reported superiority on smaller instances in [10] (up to 120 activities), our experiments with the set of test instances in [23] clearly shows that LRH has difficulty in finding feasible solutions on a significant percentage of instances. Careful analysis suggests that the test instances used in [10] have a much looser deadline and smaller duality gap compared with those of [23]. Since the Lagrangian relaxation solution may not be close to the optimal solution for the hardest cases, it is not surprising that the simple forward list scheduling heuristic failed.

We present in this paper a Lagrangian Relaxation based Forward-Backward Improvement heuristic (LR-FBI). The key improvements over LRH include: ***(i)*** the Lagrangian relaxation solution is perturbed to search more neighbours; ***(ii)*** the deadline infeasible solution is improved by the iterative forward/backward scheduling technique commonly used by meta-heuristics [12].

We compare LR-FBI with the state-of-the-art meta-heuristics [23] and CP solver [19] on a comprehensive set of test data. Our results show that LR-FBI is highly competitive especially for larger instances. We embed LR-FBI in the state-of-the-art lazy clause generation solution to further improve the performance.

2 Lagrangian relaxation based forward-backward improvement heuristic

We relax the resource constraints (3) as in [14,4] by introducing multipliers λ_{kt}, $k \in R$, $t = 0, \cdots, T$, and get the Lagrangian Relaxation Problem (LRP)

$$Z_{LR}(\lambda) = \text{maximise } LRP_\lambda(s), \quad \text{s.t.} \quad (2), (4) \tag{5}$$

where $LRP_\lambda(s) = \sum_{j \in J} rc_j(s_j) + \sum_{k \in R} \sum_t \lambda_{kt} R_k$ with

$$rc_j(s_j) = c_j e^{-\alpha(s_j + p_j)} - \sum_{t=s_j}^{s_j+p_j-1} \sum_{k \in R} \lambda_{kt} r_{jk} \tag{6}$$

The multipliers λ are iteratively updated to minimise $Z_{LR}(\lambda)$ which is an upper bound of NPV. We omit here the technical details of the Lagrangian relaxation method for RcpspDc which can be found in [4].

The solution s to $Z_{LR}(\lambda)$ is normally not feasible with respect to the resource constraints. Previously [4] we used a simple heuristic to construct a feasible solution from s, but this often fails for problems with tight deadline.

Algorithm 1: FBI(s)

```
1  best_NPV = −∞; generate keys K(s);
2  for x ∈ K(s) do
3      right = true; s' = SGS_left(x) % decode x to schedule s';
4      while makespan(s') > T do
5          if right then
6              s'' = SGS_right(s' + p) % rightmost schedule using activity end times;
7          else s'' = SGS_left(s') % leftmost schedule using activity start times;
8          if makespan(s'') ≥ makespan(s') then return best_NPV;
9          right = ¬right; s' := s'';
10     s' = shift(s'); if NPV(s') > best_NPV then best_NPV = NPV(s') ;
11 return best_NPV ;
```

For LR-FBI we try to find a feasible schedule similar to s using FBI(s) detailed in Algorithm 1. Firstly a set of keys $K(s)$ is created for s. The key is a vector $x \in \mathcal{R}^{|J|}$ which is decoded into a schedule by a Schedule Generation Scheme (SGS) [6]. The iterative forward/backward scheduling technique is used to reduce the makespan of a deadline infeasible schedule. Finally, the NPV of the schedule is further improved by shifting activities (shift) as in [10].

To calculate keys $K(s)$, rather than use a Linear Programming relaxation of the original problem [18,3,5], we use the computationally more efficient α-point idea of [14] which is based on a single LRP solution. The j^{th} key element of the m^{th} key is defined as $x_j^m = s_j + \alpha_j^m \times p_j$, $\alpha_j^m \in [0, 1]$. We have two different strategies to create $K(s)$. Best-$\alpha(k)$ generates k uniformly distributed keys with $\alpha_j^m = m/k$, $m = 0, \cdots, k-1$. Random-$\alpha(k)$ generates k random keys where each α_j^m is randomly chosen with a uniform distribution.

SGS_left(x) (SGS_right(x)) [2] greedily schedules activities one by one as early (late) as possible respecting the (reverse) precedence constraints and resource constraints, in the order where i is scheduled before (after) j if $x_i < x_j$. The resulting schedules are left(right)-justified [22]. SGS can be implemented in both serial and parallel modes [6].

3 Constraint Programming Hybrid Approach

LR-FBI can quickly find high quality solutions, but might converge to a local optima. Therefore we also investigate the possibility to further improve the solution quality using CP technology. The state-of-the-art complete method for RCPSPDC [19] is a constraint solver based on lazy clause generation [16]. Compared to the conceptual model on page 1, each resource constraint (3) is modeled by the global constraint `cumulative`$(s, p, r_{.k}, R_k)$ $(k \in R)$ where the start times variables s_i $(i \in J)$ are finite domain variables with an initial domain of $\{0, 1, \ldots, T\}$. As filtering algorithms for `cumulative`, the explanation-based version of the Time-Tabling [21] and Time-Tabling-Edge-Finder [20] are used.

In addition, the CP model uses the constraint `max_npv_prop`(s, p, c, L, NPV), recently proposed in [19], in order to compute a tight upper bound on NPV function and filter impossible values from the start times domains. This constraint considers the subproblem of RcpspDc in that (only) the resource constraints are relaxed, with the current bounds of the start times variables. Since this subproblem is polynomial solvable in time, the corresponding propagator computes its maximal NPV value, which is a valid upper bound for the original RcpspDc, and, then, uses this NPV value for tightening the bounds on the NPV function and filtering the start times.

In [19], we compared different search strategies. Here, we consider the best-performing one Vsids and combine this strategy with a Luby restart policy [13] having a restart base of 100.

The hybrid solution (CP-LR) runs after LR-FBI. Each solution s found by LR-FBI is stored in a set S. This immediately gives a much stronger lower bound for NPV. A two phase search strategy is then used. In the first phase the variable selected is the one with the largest average reduced cost defined as $\tilde{v}_j = \sum_{s \in S} rc_j(s_j)/|S|$. For value selection we maintain a reduced time window for each activity with window left (right) end defined as $w_j^l = \min_{s \in S} s_j$ ($w_j^r = \max_{s \in S} s_j$). The minimum feasible value in this reduced window will be chosen first. If no feasible value exists we rerun LR-FBI using the current bounds on the start times from the CP search to expand this window, using a limit of at most 5 iterations of LR, and adding any new solutions found to S. Our earlier work on Rcpsp [21] shows that pure Vsids search is quite robust, but can be improved using some more problem specific heuristics first. The same thing applies here. Vsids is important for robustness, but the first phase helps find good solutions earlier, and set up Vsids to be most productive. After one third of the time is used we swap to the second search phase which is pure Vsids search.

4 Experiments

We carried out extensive experiments on the benchmark set available at `www.projectmanagement.ugent.be/npv.html`. The benchmark set consists of 17280 RcpspDc instances which are split in 4 problem sizes, *i.e.*, 25, 50, 75 and 100 activities. A more detailed specification of these instances can be found in [23]. The NPV and CPU time for each instance is also available for the scatter search method in [23] which terminates when a maximal number of schedules are generated using a computer with a Dual Core processor 2.8GHz.

The parameter settings of LR can be found in [4]. We implemented our CP based approach using the Lcg solver Chuffed. All tests were run on a computing cluster of which each node has two 2.8GHz AMD 6-Core CPU. We used a time limit of 5 minutes. The time limit for search in CP-LR is also 5 minutes.

We illustrate the effects of our improvements on the LRH in [10] in Table 1. We report the percentage of instances which have feasible schedule found (Fea%), the number of instances on which the best NPV is achieved (Best) and the average relative deviation (Dev) on instances for which all methods find a

Table 1. Comparison of feasibility results on 100-activities instances

	Fea%	Dev_V	Dev	Best
LRH	72.9	203.3	76.1	836
S-Best-α(1)	77.3	203.0	74.7	1005
P-Best-α(1)	91.9	207.6	79.5	450
P-Best-α(10)	95.8	197.1	69.4	1919
P-Random-α(2000)	99.5	197.1	69.4	2304

Table 2. Comparison of Scatter search, CP and LR

	Scatter(5000)			CP-Vsids			P-Random-α(2000)			CP-LR		
size	Fea%	Dev	Best	Fea%	Dev	Best	Fea%	Dev	Best	Fea%	Dev	Best
25	100.0	73.1	2507	100	72.8	3663	99.8	81.6	795	100.0	72.3	3708
50	99.9	91.6	1556	98.4	104.8	1124	99.9	82.4	937	100.0	78.6	2345
75	99.7	106.9	1196	90.3	-	-	99.8	98.3	1836	99.98	97.4	3054
100	99.6	100.2	1612	76.8	-	-	99.5	95.3	1524	99.7	93.6	2641

Table 3. Comparison with best scatter search results on size 100

	Fea%	Dev	Best	Ave(s)	Max(s)
Scatter(50000)	99.6	89.9	2003	26.2	139.8
P-Random-α(2000)	99.5	86.5	1283	167	607
CP-LR	99.8	85.4	2240	300	300

feasible schedule. Dev is defined as [23] $abs((Ub - Lb)/Ub)$. Dev_V is calculated with the upper bound in [23], while Dev uses the LR upper bound. The prefix S-(P-) stands for the serial (parallel) SGS. It can be seen that LRH has serious problems with feasibility. Parallel SGS is superior to serial SGS in terms of feasibility. The use of α-point further improves both feasibility and NPV. Since LR can produce much stronger upper bounds than [23] we only use Dev for the remaining tests.

We compare the reported results for scatter search [23] with at most 5000 schedules, with CP [19], LR-FBI and our hybrid CP approach (CP-LR) in Table 2. Scatter search is very fast (average computation time is 4.2s for size 100) and almost always finds a feasible solution. CP performs very well on the smallest instances but does not scale well. LR-FBI is highly competitive when problem sizes increase, generally finding better solutions, but requires more time than scatter search (82s on average for size 100). Running the hybrid CP-LR substantially improves on LR-FBI on a large number of instances. Clearly the hybrid is much more robust than a pure CP approach.

We also compare with the best results of scatter search with 50000 schedules in Table 3, showing also average and maximum solving time. Scatter search reduces the deviation by 10% with significant increase of solution time. The time limit for LR-FBI is set to 10 minutes. The LR-FBI has better deviation than that of scatter search. The hybrid method CP-LR improves the LR-FBI results further, resulting in the best known results on these instances.

5 Conclusion

We developed a new Lagrangian relaxation based heuristic for the RcpspDc problem and achieved highly competitive results on a comprehensive set of test data. We also investigated the integration of our heuristic into a CP solver and obtained promising results. We have built an effective hybrid of local search and complete search, by using local search information not only for bounding but to direct the initial search phase. This hybrid is interesting since it runs the local search *on demand* when it can no longer provide useful guidance to the complete search. Our future research will focus on more efficient hybridisation of LR, CP and meta-heuristics for the RcpspDc problem.

Acknowledgements NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

1. Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research 112(1), 3–41 (1999)
2. Debels, D., Vanhoucke, M.: A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. Operations Research 55(3), 457–469 (2007)
3. Gu, H.Y.: Computation of approximate alpha-points for large scale single machine scheduling problem. Computers & OR 35(10), 3262–3275 (2008)
4. Gu, H., Stuckey, P.J., Wallace, M.G.: Maximising the net present value of large resource-constrained projects. In: Milano, M. (ed.) Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming. LNCS, vol. 7514, pp. 767–781. Springer (2012)
5. Gu, H., Xi, Y., Tao, J.: Randomized Lagrangian heuristic based on Nash equilibrium for large scale single machine scheduling problem. In: Proceedings of the 22nd IEEE International Symposium on Intelligent Control. pp. 464–468 (2007)
6. Hartmann, S., Kolisch, R.: Experimental evaluation of state-of-the-art heuristics for resource constrained project scheduling. European Journal of Operational Research 127, 394–407 (2000)
7. Hartmann, S., Briskorn, D.: A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207(1), 1–14 (2010)
8. Herroelen, W.S., Demeulemeester, E.L., De Reyck, B.: A classification scheme for project scheduling. In: Weglarz, J. (ed.) Project Scheduling, International Series in Operations Research and Management Science, vol. 14, pp. 1–26. Kluwer Academic Publishers (1999)
9. Icmeli, O., Erengüç, S.S.: A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. Management Science 42(10), 1395–1408 (1996)

10. Kimms, A.: 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 (2001)
11. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, 23–37 (2006)
12. Li, K., Willis, R.: An iterative scheduling technique for resource-constrained project scheduling. European Journal of Operational Research 56, 370–379 (1992)
13. Luby, M., Sinclair, A., Zuckerman, D.: Optimal speedup of Las Vegas algorithms. Inf. Proc. Let. 47(4), 173 – 180 (1993)
14. Möhring, R.H., Schulz, A.S., Stork, F., Uetz, M.: Solving project scheduling problems by minimum cut computations. Management Science 49(3), 330–350 (2003)
15. Neumann, K., Zimmermann, J.: Exact and truncated branch-and-bound procedures for resource-constrained project scheduling with discounted cash flows and general temporal constraints. Central European Journal of Operations Research 10(4), 357–380 (2002)
16. Ohrimenko, O., Stuckey, P.J., Codish, M.: Propagation via lazy clause generation. Constraints 14(3), 357–391 (2009)
17. Russell, A.H.: Cash flows in networks. Management Science 16(5), 357–373 (1970)
18. Savelsbergh, M., Uma, R., Wein, J.: An experimental study of LP-based approximation algorithms for scheduling problems. INFORMS J. on Computing 17, 123–136 (2005)
19. Schutt, A., Chu, G., Stuckey, P.J., Wallace, M.G.: Maximising the net present value for resource-constrained project scheduling. In: Beldiceanu, N., Jussien, N., Pinson, E. (eds.) Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems. LNCS, vol. 7298, pp. 362–378. Springer (2012)
20. Schutt, A., Feydy, T., Stuckey, P.J.: Explaining time-table-edge-finding propagation for the cumulative resource constraint. In: Gomes, C.P., Sellmann, M. (eds.) Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems. LNCS, vol. TBA, p. TBA. Springer (2013)
21. Schutt, A., Feydy, T., Stuckey, P.J., Wallace, M.G.: Explaining the cumulative propagator. Constraints 16(3), 250–282 (2011)
22. Sprecher, A., Kolisch, R., Drexl, A.: Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research 80, 94–102 (1995)
23. Vanhoucke, M.: A scatter search heuristic for maximising the net present value of a resource constrained project with fixed activity cash flows. International Journal of Production Research 48(7), 1983–2001 (2010)
24. Vanhoucke, M., Demeulemeester, E.L., Herroelen, W.S.: On maximizing the net present value of a project under renewable resource constraints. Management Science 47, 1113–1121 (Aug 2001)
25. Zhu, D., Padman, R.: A metaheuristic scheduling procedure for resource-constrained projects with cash flows. Naval Research Logistics 46, 912–927 (1999)