A genetic approach based recovering beam search for minimizing the total completion time in a single machine scheduling problem subject to release dates

Abstract

In this paper we consider the problem of minimizing the sum of completion times in a single machine scheduling problem subject to release dates. We propose a new hybrid heuristic by combining a genetic local search algorithm and a recovering beam search procedure. Computational experiments carried out on a large set of random instances show that the proposed hybrid method consistently yields optimal or near-optimal solutions. The heuristic produces solutions that outperform the best known ones.

Introduction

Consider n jobs which are to be scheduled without preemption on a single machine available from time 0 onwards. Each job j is defined by a release time r_j and a processing time p_j. The aim is to determine a job sequence which minimizes the sum of completion times, denoted by $\sum C_j$ with C_j the completion time of job j. This problem can be referred to as $1 \mid r_j \mid \sum C_j$. In this paper we propose a new hybrid heuristic by combining a genetic local search algorithm (Rakrouki and Ladhari 2008) and a recovering beam search procedure (Della Croce et al. 2004).

Literature review

The $1 \mid r_j \mid \sum C_j$ scheduling problem is known to be $\mathcal{NP}$-hard in the strong sense (Rinnooy Kan 1976). With equal release dates the problem is solved in polynomial time by SPT priority rule (Smith 1956). The preemptive version of the problem, referred to as $1 \mid r_j, pmtn \mid \sum C_j$, is optimally solvable in $O(n \log n)$ time with the SRPT priority rule (Baker 1974).

The $1 \mid r_j \mid \sum C_j$ problem has been extensively studied in the literature. Several approximate methods have been proposed for this problem. Liu and MacCarthy (1991) proposed some heuristics for the problem and gave conditions under which the heuristics provide optimal sequences. Reeves (1995) generalized the Liu and MacCarthy (1991) heuristics. Chu (1992a) have presented two priority rules for total flowtime (PRTF and APRTF). Chand, Traub, and Uzsoy (1996b) have proposed a polynomial ($O(n^4 \log n)$) iterative algorithm. Also, a rolling horizon method have been proposed by Chand, Traub, and Uzsoy (1997). Della Croce and T'kindt (2002) have provided excellent results by developing a powerful Recovering Beam Search procedure with an $O(n^3 \log n)$ time complexity. This heuristic outperforms both in terms of quality and computing time the state of the art heuristic procedure of Chand, Traub, and Uzsoy (1996b). Guo et al. (2004) have provided a study of this problem based on extensive experiments. The objective is to study experimentally how well, on the average, this problem can be solved. They evaluated average solution quality in order to identify the characteristics of difficult instances. They also developed an empirical model to predict solution quality.

Recently, Jouglet et al. (2008) have developed dominance-based heuristics for several objective function including total completion time. They described dominance rules for these criteria, as well as techniques for using these dominance rules to build heuristic solutions. Also, they introduced a Tabu Search method with a neighborhood based on these dominance rules.

Several branch-and-bound algorithms and/or dominance properties have been designed for the $1 \mid r_j \mid \sum C_j$ problem (see Chu 1992b; Chand, Traub, and Uzsoy 1996a; Jouglet et al. 2008). Ahmadi and Bagchi (1990) have proven that the optimal solution of $1 \mid r_j, pmtn \mid \sum C_j$ given by SRPT is the best available lower bound for the $1 \mid r_j \mid \sum C_j$ problem. Della Croce and T'kindt (2003) have proposed an improvement on SRPT lower bound by exploiting the properties of the preemptive solution. The proposed improvement reduces by approximately 44% on the average the gap between the preemptive solution value and the optimal solution value. Chu (1992b) has proposed a branch-and-bound algorithm using some efficient dominance rules which can handle instances with up to 100 jobs. Several dominance properties are provided. Chand, Traub, and Uzsoy (1996a) also proposed an exact method which provide similar results than those of Chu. Currently, the best known exact method is the branch-and-bound algorithm of T'kindt et al. (2004) which can solve problems with up to 130 jobs.

Recovering beam search algorithm

Beam search (BS) is a heuristic method that was introduced in the context of scheduling by Ow and Morton (1988). BS is a derivative of branch-and-bound algorithm in which only

some nodes are explored. It thus requires less computational time, but can no longer guarantee an optimal solution. An improvement of BS, named Recovering Beam Search (RBS), have been proposed by Della Croce et al. (2004) based on the following remark: if a branch leading to the optimal solution in the search tree is pruned in the nodes evaluation process, there is no way to reach afterwards that solution. In RBS algorithm, a two-stage approach is applied at each level for choosing the most promising nodes. A first fast evaluation (*filter*) is applied to choose φ nodes (called *filter width*). Then, the selected nodes are accurately evaluated and the best w (called *beam width*) nodes are retained for branching. Each node is evaluated by means of a weighted sum of lower and upper bounds. Once the best w nodes and the corresponding best partial solution are retained, a recovering step is applied to check whether the current partial solution x is dominated by another partial solution y having the same level of the search tree. This recovering subroutine generates for each node, several feasible solutions and the best among them represents the node upper bound. RBS have been successfully applied on the $1 \mid r_j \mid \sum C_j$ problem by Della Croce and T'kindt (2002).

Let w a fixed beam width, σ the current partial solution, Q the set of partial solutions to branch from and z the best solution value. The main steps of the RBS procedure are the following:

Algorithm 1 RBS algorithm

Step 1: *Initialization*
$\sigma = \emptyset; Q = \emptyset; z = +\infty;$
Step 2: *Branching*
Branch from all σ in Q generating the corresponding children. Select all branches that are not dominated by means of available problem-specific dominance properties or eliminated by the filter.
For each selected branch **Do**
Compute lower and upper bounds, LB and UB, for that branch.
If $UB < z$ **Then** $z = UB$.
Compute the evaluation function $V = (1 - \alpha)LB + \alpha UB$ with $0 \leq \alpha \leq 1$.
End for
Step 3: *Selection*
Set Q to the w partial solution σ^* with lowest value of V.
Step 4: *Recovering*
For all $\sigma \in Q$, search for a partial solution σ' that dominates σ by means of interchange operators.
If σ' is found **Then** replace σ by σ' in Q.
Step 5: *Termination condition*
If $Q \neq \emptyset$ **Then** go to **Step 2 Else** Stop and z is the final solution value.

In the RBS of Della Croce and T'kindt (2002) the SRPT rule is used to compute a valid lower bound LB whilst the upper bound UB is computed using the PRTF and APRTF rules of Chu (1992a). In the recovering step some dominance properties are used. The RBS algorithm has a time complexity which linearly depends on the value of the beam width w. Della Croce and T'kindt (2002) considered $w = 1$.

Genetic local search algorithm

In this section we give a brief description of the main components of the genetic local search algorithm (GLS) of Rakrouki and Ladhari (2008).

The initial population consists of M ($M = 200$) permutations. $M - 1$ chromosomes are generated randomly while a single chromosome is generated using a constructive heuristic (Rakrouki and Ladhari 2008). With a probability P_C ($P_C = 0.9$), we used three different crossover operators: Two versions of the two-point crossover (used with a probability $P_{C1} = P_{C2} = 0.3$) and the three-point crossover (used with a probability $P_{C3} = 0.4$). The mutation step is used with a probability P_{Mu} ($P_{Mu} = 0.7$). Two mutation operators are adopted: The random exchange mutation and the insertion mutation (used with a probability $P_{Mu1} = P_{Mu2} = 0.5$). As termination condition, the GLS stops either when a maximum number of $100 \times n$ generation has been reached or when the best solution of the population has not been improved after $10 \times n$ consecutive generations. After the mutation step, we apply local search phase, in order to derive high quality solutions, using two procedures with a probability $P_{LS} = 0.5$. The first procedure generates k ($k = 30$) neighbors for each solution in the population using random exchanging and insert the best to the population. The second procedure improves each solution in the population by means of random exchanging and adjacent swapping.

The main steps of the GLS algorithm are the following:

Algorithm 2 Genetic Algorithm

Step 0: *Initialize control parameters*
Step 1: *Generation of the initial population*
Generate the initial population.
Compute the total completion time of each solution.
Step 2: *Improvement of the population*
Select two parents using a binary tournament strategy.
Apply crossover operator on the two parents according to the probability P_C.
Apply Mutation operator on the generated offspring with a probability P_{Mu}.
Step 3: *Evaluation and selection*
Evaluate the generated offspring.
Replace the worst chromosome in the population by the new generated offspring if the fitness of this later is better.
If the number of generated offsprings are less than M **Then** apply local search procedures with a probability P_{LS} and go to **Step 2**.
Step 4: *Termination condition*
If the termination condition is satisfied **Then** stop **Else** go to **Step 2**.

Hybrid recovering beam search algorithm

In general, RBS algorithms have the advantage that they are strongly guided by the deterministic use of a weighting evaluation function. However, the use of a deterministic policy for extending partial solutions may reduce, in case the weighting function is bad, the chances of finding high-quality solutions. On the other side, in the GLS algorithm randomization often helps in searching around presumably good solutions. However, the method is highly sensible to the balance between heuristic guidance and randomization. Based on these considerations we expect a benefit from combining these two ways of exploring a search space.

We propose a new hybrid recovering beam search (HRBS) procedure by embedding a variant of the genetic local search algorithm (GLS) of Rakrouki and Ladhari (2008) into a modified version of the RBS procedure of Della Croce and T'kindt (2002).

Let σ a partial solution of the search tree of the RBS procedure. The GLS algorithm is called for computing an upper bound. GLS completes randomly the partial sequence σ with the remaining jobs and inserts it in the initial population. Then, GLS is executed for computing a valid upper bound. Indeed, RBS procedure calls the GLS algorithm only at a fixed depth (*beam depth*) of the search tree and for each node of the w selected nodes (*beam width*). Also, the node evaluation process is modified in the RBS by eliminating the first stage (*filter*). The two parameter values (i.e. beam depth and beam width) of the RBS procedure are chosen according to the problem size (n). Both are fixed experimentally to $\frac{n}{4}$.

In order to minimize the CPU time required by the HRBS procedure, we reduce some parameters of GLS. The maximum number of generations is fixed at $n \times 10$ and the maximum of consecutive generations without improving the best solution of the population is fixed at $n \times 5$.

Computational results

This section describes the computational tests which have been conducted to evaluate the empirical performance of the proposed algorithms. These procedures have been coded in *C* and the computational experiments were carried out on a Pentium *IV 3.0 GHz PC with 1GB RAM.*

Tests problems

In order to be able to compare our results to previous relevant experimental studies, the instances are the same as test problems of Jouglet et al. (2008). These instances are generated using the scheme provided by Hariri and Potts (1983).

For each job, the processing time is randomly generated from the uniform distribution in $[1, 100]$. For a problem size $n \in \{10, 20, ..., 100\}$, an integer release date for each job was generated from the uniform distribution $[0, 50.5nR]$, where R controls the range of the distribution. For each selected value of n, 20 problems were generated for each of the R values 0.2, 0.4, 0.6, 0.8, 1.0, 1.25, 1.5, 1.75, 2.0 and 3.0 producing 200 problems for each value of n. Thus, a total of 2000 instances were generated.

Performance of the proposed algorithms

The performance analysis is based on three measures: the average relative gap from the optimal solution ($gapOpt$) and percentage number of times an algorithm reaches the optimal solution ($OptS$) on problems with up to 100 jobs. Also, we provide the results of the TS (Tabu Search) heuristic of Jouglet et al. (2008) and the RBS of Della Croce and T'kindt (2002).

The relative gap provides the gap between the solution given by an algorithm and the optimal solution ($gapOpt$) on problems with up to 100 jobs (i.e. $gapOpt = (\frac{UB - OPT}{OPT}) \times 100$ where UB is the solution value of one of the proposed heuristics and OPT is the optimal solution value was computed by means of the branch-and-bound algorithm kindly provided by Jouglet et al. (2008).

The results of the computational study of the proposed algorithms are summarized in Tables 1–3. These tables provide average relative gap from the optimal solution value ($gapOpt$), percentage number of times an algorithm reaches the optimal solution ($OptS$) and average CPU time ($Time$, in seconds).

n	*GLS*	*RBS*	*HRBS*	*TS*
10	0	0.0122	0	0
20	0	0.0116	0	0
30	0	0.0246	0	0
40	0.0012	0.0203	0.0003	0.0020
50	0.0014	0.0189	0.0005	0.0040
60	0.0012	0.0171	0.0016	0.0120
70	0.0022	0.0187	0.0010	0.0220
80	0.0024	0.0196	0.0015	0.1340
90	0.0031	0.0135	0.0013	0.3280
100	0.0029	0.0141	0.0025	0.9910
avg.	***0.0014***	***0.0171***	***0.0009***	***0.1493***

Table 1: The average gap to optimality

From Table 1, we remark that HRBS outperforms both TS and RBS for all the problem sizes. Also, HRBS provides better average relative gap than GLS for all problem sizes except for $n = 60$.

n	*GLS*	*RBS*	*HRBS*	*TS*
10	100	96	100	100
20	100	91.5	100	100
30	100	71	100	100
40	97.5	66	98.5	99
50	93.5	60.5	95	98
60	92.5	57.5	92.5	98
70	90.5	47	93	94
80	88.5	36.5	87	80
90	83	42.5	84	70
100	76.5	38.5	77	60
avg.	***92.2***	***60.7***	***92.7***	***89.9***

Table 2: The percentage of optimal solution found

Table 2 shows that HRBS reaches the optimum for 100% of the instances for problem sizes up to 30 jobs. Also,

GLS and HRBS have usually nearly the same percentage of instances solved optimally. Moreover, we remark that HRBS is able to solve instances optimally more than TS for medium problem sizes ($n \geq 80$).

n	***GLS***	***RBS***	***HRBS***	***TS***
10	0.24	0.00	*0.20*	0.00
20	0.94	0.00	*1.10*	0.00
30	2.03	0.01	*4.20*	0.00
40	3.96	0.01	*13.73*	0.27
50	4.80	0.02	*29.74*	1.07
60	7.67	0.02	*66.75*	2.98
70	12.78	0.04	*135.57*	5.80
80	19.70	0.06	*245.32*	9.19
90	29.77	0.09	*404.41*	12.42
100	43.39	0.09	712.87	16.20
avg.	***12.53***	***0.03***	***161.39***	***4.79***

Table 3: CPU time required by the algorithms on small and medium size problems (*Time*)

Table 3 reveals that HRBS requires more CPU time than the other presented heuristics: all the instances were solved by HRBS within a mean CPU time nearly equal to $3\,\mathrm{min}$, which remains reasonable.

We observe from Table 1, that RBS outperforms TS for $n \geq 70$. However, for problem sizes with $n \leq 70$, TS performs better than both HRBS, GLS and RBS in term of percentage of instances solved to optimality, but not regarding the average gap to optimality. From these experimental results we can evince that HRBS outperforms the existing state-of-the-art heuristics.

Conclusion

This paper investigates the problem of minimizing of the total completion time subject to release dates on a single machine environment. For this classical $\mathcal{NP}$-hard problem we proposed a new hybrid recovering beam search procedure. Computational experiments provide strong evidence that the proposed algorithm outperforms the best known heuristics of the literature.

References

Ahmadi, R.H., and Bagchi, U. 1990. Lower bounds for single-machine scheduling problems. *Naval Research Logistics* 37: 967–979.

Baker, K.R. eds. 1974. *Introduction to Sequencing and Scheduling*. Wiley Publishing, New York.

Chand, S., Traub, R., and Uzsoy, R. 1996a. Single-Machine Scheduling with Dynamic Arrivals: Decomposition Results and an Improved Algorithm. *Naval Research Logistics* 43: 709–716.

Chand, S., Traub, R., and Uzsoy, R. 1996b. An iterative heuristic for the single-machine dynamic total completion time scheduling problem. *Computers and Operations Research* 23: 641–651.

Chand, S., Traub, R., and Uzsoy, R. 1997. Rolling horizon procedures for the single machine deterministic total completion time scheduling problem with release dates. *Annals of Operations Research* 70: 115–125.

Chu, C. 1992a. Efficient heuristics to minimize total flow time with release dates. *Operations Research Letters* 12: 321–330.

Chu, C. 1992b. A branch-and-bound Algorithm to Minimize Total Flow Time with Unequal Release Dates. *Naval Research Logistics* 39: 859–875.

Della Croce, F., and T'kindt, V. 2002. A recovering beam search algorithm for the one-machine dynamic total completion time scheduling problem. *Journal of Operational Research Society* 53: 1275–1280.

Della Croce, F., and T'kindt, V. 2003. Improving the preemptive bound for the one-machine dynamic total completion time scheduling problem. *Operations Research Letters* 31: 142–148.

Della Croce, F., Ghirardi, M., and Tadei, R. 2004. Recovering Beam Search Approach for Combinatorial Optimization Problems. *Journal of Heuristics* 10: 89–104.

Guo, Y., Lim, A., Rodrigues, B., and Yua S. 2004. Minimizing total flow time in single machine environment with release time: an experimental analysis. *Computers & Industrial Engineering* 47: 123–140.

Hariri, A.M.A., and Potts, C.N. 1983. An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Applied Mathematics* 5: 99–109.

Jouglet, A., Savourey, D., Carlier, J., and Baptiste, P. 2008. Dominance-based heuristics for one-machine total cost scheduling problems. *European Journal of Operational Research* 184: 879–899.

Liu, J., and MacCarthy, B.L. 1991. Effective heuristics for the single machine sequencing problem with ready times. *International Journal of Production Research 29*: 1521–1533.

Ow, P.S., and Morton, T.E. 1988. Filtered beam search in scheduling. *International Journal of Production Research* 26: 297–307.

Rakrouki, M.A., and Ladhari, T. 2008. Approximate procedures for minimizing total completion time in a single machine scheduling problem subject to release dates. The Eleventh International Workshop on Project Management and Scheduling (PMS2008), 218-221. Bogaziči University, Istanbul, Turkey.

Reeves, C. 1995. Heuristics for scheduling a single machine subject to unequal job release times. *European Journal of Operational Research* 80: 397–403.

Rinnooy Kan, A.H.G. 1976. *Machine sequencing problem: classification, complexity and computation*. Nijhoff, The Hague.

Smith, W.E. 1956. Various Optimizers for Single Stage Production. *Naval Research Logistics Quarterly* 3: 59–66.

T'kindt, V., Della Croce, F., and Esswein, C. 2004. Revisiting Branch and Bound Search Strategies for Machine Scheduling Problems. *Journal of Scheduling* 7(6): 429–440.