LNCS manuscript No.
(will be inserted by the editor)

Hybrid Heuristics for Dynamic Resource-Constrained Project Scheduling Problem

André Renato Villela da Silva · Luiz Satoru Ochi

Received: date / Accepted: date

Abstract Dynamic Resource-Constrained Project Scheduling Problem (DRCPSP) is a scheduling problem that works with an uncommon kind of resources: the Dynamic Resources. They increase and decrease in quantity according to the activated tasks and are not bounded like other project scheduling problems. This paper presents a new mathematical formulation for DRCPSP as well as two hybrid heuristics merging an evolutionary algorithm with an exact approach. Computational results show that both hybrid heuristics present better results than the state-of-the-art algorithm for DRCPSP does. The proposed formulation also provides better bounds.

Keywords mathematical formulation · hybrid heuristics · project scheduling problems · dynamic resources

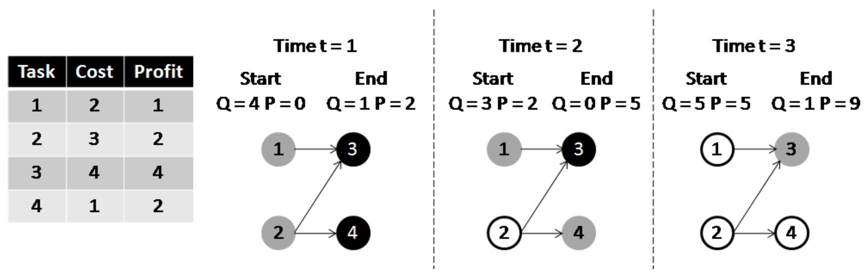
1 Introduction

The Dynamic Resource-Constrained Project Scheduling Problem deals with an uncommon kind of resource called Dynamic Resource. Unlike classical project scheduling problems [1,2,6] where resources may be renewable or nonrenewable (both with a bounded quantity), the DRCPSP allows an unbounded amount of these resources because they are consumed when a task is executed, but are produced by tasks after their activations.

DRCPSP can be described as follows. Let $G = (V, A)$ be a directed acyclic graph (DAG), where V is the set of vertices and A is the set of arcs that represent the tasks (activities) and their precedences, respectively. There are an activation cost c_i and a profit p_i (positive integer values) associated to each task i . There are also a planning horizon represented by a time interval composed of H time units, an amount of initial

This work was partially supported by CNPq - grant 141074/2007-8

Universidade Federal Fluminense, Instituto de Computação
Rua Passo da Pátria, 156 – Bloco E – 3^o andar – Boa Viagem
24210-240, Niterói, RJ, Brazil
Tel./Fax: +55-21-26295669
E-mail: {avillela,satoru}@ic.uff.br



Task	Cost	Profit
1	2	1
2	3	2
3	4	4
4	1	2

Fig. 1 Example of a scheduling in DRCPSP

resources Q_0 and an initial accumulated profit $P_0 = 0$. Variables Q_t and P_t represent the available resources and the accumulated profit at time t.

When a task i is activated at time t, it is necessary to pay its cost retrieving c_i from Q_t. From the end of time t to the end of time H (planning horizon size), p_i units of resources are generated by activated task i at each of these time units. In other words, p_i units are added to the accumulated profits from t to H. When the scheduling moves to the next time unit, all accumulated profit becomes available resources and can be used to activate more tasks. The objective of DRCPSP is to maximize the resources $(Q_H + P_H)$ at the end of time H. Basically, two constraints are imposed to activate a task i at time t: (i) there must be enough available resources Q_t to activate the task and (ii) a task may be activated only if all predecessors of i are activated before t.

Fig. 1 shows an example of a scheduling, supposing $H = 3$, $Q_0 = 4$, $P_0 = 0$. For the sake of convenience, we will use a simplified notation of Q and P. At the start of time $t = 1$ there are two available tasks 1 and 2 (in gray). Task 2 is scheduled first. We need to retrieve 3 units of resource from Q (available resources) and add 2 units to P (profits earned so far). Moving to the next time $t = 2$, profit P is added to available resources Q. Task 2 is already scheduled (in white) and tasks 1 and 4 are available and will be scheduled. We finish this time unit without any resources ($Q = 0$), but profit $P = 5$. Moving to the next time, we have $Q = 5$, $P = 5$ and only task 3 is available. We schedule it and finish time unit $t = 3$ with $Q = 1$ and $P = 9$. Our scheduling finishes and the value of the final solution is given by the sum of $Q = 1$ and $P = 9$ (10 units of resource).

DRCPSP is a NP-hard problem and has application in commercial and industrial environments where expansion plans might take place. Suppose a company wishes to open new branches in order to increase its financial gains. Each candidate city or place has a building cost related and it is supposed to obtain some sustained profit after its opening. Due to logistic issues (warehouses capacity, transport services or other management issues) it is not possible to open branches anywhere, anytime, therefore branches must be opened with some precedences. In general, a fixed budget is available to the expansion plan and all other monetary resources must be acquired by the business profits. Expansion plans like this have a time period to be projected and executed and the main objective is to obtain the maximum amount of monetary resources as possible at the end of the planning horizon.

The remainder of this paper is organized as follows. Section 2 provides a brief description of literature algorithms and their benefits. Section 3 presents a new mathematical formulation for DRCPSP as a Mixed Integer Program (MIP) as well as two new hybrid heuristics. Section 4 shows the computational results. Finally, the conclusion remarks are presented in Section 5.

2 Literature Review

DRPSP was originally proposed in [10]. Two Evolutionary Algorithms (EA1 and EA2) and a mathematical formulation were developed in this work. The formulation consists in a MIP that will be explained in the next section. This first formulation will be called F1 from now on.

A better Evolutionary Algorithm (EA3) proposed in [11] could overcome the problems of EA1 and EA2 using a reconstruction scheme that tries to avoid premature convergence. In the same paper a first hybrid heuristic using CPLEX and EA3 was also proposed. It divided the planning horizon in two halves. CPLEX was used in the first half and the partial solution was given to EA3 to finish the second half. This hybrid heuristic achieved good results when planning horizon was short, but consumed very long time when a larger horizon was considered - CPLEX's half scheduling took too much time to finish.

A new representation for DRCPSP solutions have been adopted since then [12,13]. This representation handles priorities for the scheduling of tasks in a similar way as the random-keys approach [5] does. A task with higher priority should be scheduled first if no constraints are violated. A new scheduling algorithm in conjunction with the representation allows that any priority list produces a feasible scheduling solution.

New EA versions were proposed exploiting this feature. The first EA that uses this indirect representation is called EA_priority. The initial population is computed giving to each task of each individual a priority based on the profit p_i and the cost c_i. The population is ranked according to the fitness of individuals and the population is divided into three classes: A (best individuals), B and C (worst individuals). A parent from class A and another one from class B are randomly chosen and the average of parents priorities are computed and given to the offspring. A mutation operator might be applied to each individual, then, the best individuals are chosen to remaining alive in the next generation.

The second Evolutionary Algorithm which uses the indirect representation is called EA_ages and was proposed in [13]. EA_ages is the best heuristic known for DRCPSP and its features are going to be reviewed in Subsection 3.1.

3 New Mathematical Formulation for DRCPSP

The first mathematical formulation proposed for DRCPSP was presented in [10]. This formulation will be called F1 and is presented in a more simplified way as follows.

$$\text{(F1) Max} \quad Q_H + P_H \tag{1}$$

Subject to

$$x_{it} \leq \sum_{t'=1}^{t-1} x_{jt'} \quad \forall i = 1, ..., n \quad \forall t = 1, ..., H \quad \forall j \in Pred(i) \tag{2}$$

$$x_{i0} = 0 \quad \forall i = 1, ..., n \tag{3}$$

$$Q_t = Q_{t-1} + P_{t-1} - \sum_{i=1}^{n} c_i x_{it} \quad \forall t = 1, ..., H \tag{4}$$

$$P_t = P_{t-1} + \sum_{i=1}^{n} p_i x_{it} \quad \forall t = 1, ..., H \tag{5}$$

$$\sum_{t=1}^{H} x_{it} \leq 1 \quad \forall i = 1, ..., n \tag{6}$$

$$x_{it} \in \{0, 1\} \quad \forall i = 1, ..., n \quad \forall t = 1, ..., H \tag{7}$$

$$Q_t, P_t \in N \quad \forall t = 0, ..., H \tag{8}$$

If a task i is activated at time t, variable x_{it} is equal to one; otherwise, it is zero. The objective function (1) seeks to maximize the resources at the end of time H. Constraints (2) ensure that a task i can be activated only if all its predecessor tasks are activated previously. All tasks are inactive at the start of scheduling (3). Constraints (4) show the available resources at time t. Constraints (5) state that the accumulated profit at time t is equal to the accumulated profit at prior time plus the profit of all activated tasks at this time. Constraints (6) ensure that a task i is activated once, at most. Constraints (7) and (8) define the domain of all variables.

In medium-size instances (more than 300 tasks), F1 takes a very long time to obtain an optimal solution. In hard instances, a feasible solution was not found after more than 40 hours. The main drawback is in the relationship among the variables related to the same task. The variable x_{it}, for instance, does not provide any information about what happened to task i at time $t-1$ or earlier. So, a variable only indicates what happened in its time and no more. To know if a task was activated or not, we need to compute the sum of all variables related to this task.

In order to strengthen the relationship among the variables we propose to define a different meaning for the variables. In the second formulation, called F2, variables y_{it} indicate whether a task i was activated at time t or before that. A zero value means that the task was still not activated and an one value means that the task was already activated at that time. So, when a task is activated at time t, all variables related to that task, from time t to time H, have to be set to value one. This relationship is stronger because, now, we can look at a variable and know about an eventual past activation. Since the problem allows only one activation at most, we can know if a task was activated earlier looking at only one variable. The proposed formulation is presented and described next.

(F2) Max (1)

Subject to

$$y_{it} \leq y_{it+1} \quad \forall i = 1, .., n \quad \forall t = 1, ..., H-1 \tag{9}$$

$$y_{it} \leq y_{jt-1} \quad \forall i = 1, .., n \quad \forall t = 2, ..., H \quad \forall j \in Pred(i) \tag{10}$$

$$Q_t = Q_{t-1} + P_{t-1} - \sum_{i=1}^{n} c_i (y_{it} - y_{it-1}) \quad \forall t = 1, ..., H \tag{11}$$

$$P_t = \sum_{i=1}^{n} p_i y_{it} \quad \forall t = 0, ..., H \tag{12}$$

$$y_{i0} = 0 \quad \forall i = 1, .., n \tag{13}$$

$$y_{it} \in \{0, 1\} \quad \forall i = 1, .., n \quad \forall t = 1, .., H \tag{14}$$

(8)

Variables Q_t and P_t have the same meaning of F1 as well as the objective function (1). Constraints (13) and (14) define the mentioned relationship among variables and the precedence constraints, respectively. Constraints (15) and (16) have equivalent meaning of constraints (4) and (5), respectively.

3.1 Hybrid Heuristics

Both hybrid heuristics proposed in this paper make use of the best heuristic known for DRCPSP. This heuristic (EA_ages) is an evolutionary algorithm version with several operators and features. This algorithm was proposed in [13]. An individual is composed of a list of priorities. Tasks with higher priorities will be scheduled first if precedence and resource constraints are respected. The initial population is created giving to each individual a priority list. The priorities take into account the cost, the profit and the number of successors of each task. The population is sorted and divided into three classes. Two parents from the two highest classes are chosen and recombined through the computation of the priorities average. The mutation operator tries to improve an individual by increasing or decreasing some of its priorities.

At the end of the current generation, a natural selection takes place. The best individuals remain alive to the next generation while the other individuals are discarded. If the algorithm meets the stopping criterion, the best solution found so far will be shown. Otherwise, three cases are analyzed: (i) if the best solution was improved in the current generation, the algorithm goes back to the parent selection operator and continues; (ii) if there are 30 generations without improving the best solution, a large exploration is done. This operator consists in executing a local search around the best individuals of the current population. (iii) The third case is triggered when 4 consecutive large explorations do not improve the best solution. The whole population is discarded and a new one is created taking the best solution found so far as a seed. This reconstruction is an attempt to intensify the search near to that best solution. A framework of this evolutionary algorithm is shown in Fig. 2. All the algorithms parameters were defined after preliminary tests.

A parallel version was also proposed. In this version, some operators as Recombination and Mutation have their workload divided among the processing cores of a multi-core processor. Nowadays, these processors are very common, but a standard sequential approach uses only a part of the total processing power. The results from the parallel version shown that better solutions are found faster than the sequential version does.

Hybrid heuristics combining EAs and solvers like CPLEX have been used to obtain results closer to the optimal one or to solve NP-hard problems in a smaller amount of time. The way these approaches are combined may vary from problem to problem as can be seen in [14, 9, 7].

The first hybrid heuristic proposed in this paper merges the exact CPLEX optimizer with the evolutionary algorithm in a very common way. The evolutionary heuristic provides a solution and uses CPLEX as a local search tool in order to improve such solution. This local search, called LS_cplex, traverses the neighborhood of a solution

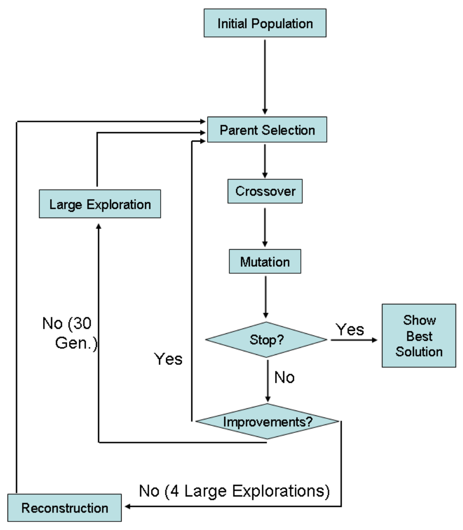


Fig. 2 The EA_ages' scheme

S similarly to the local branching technique [3,4]. Suppose that S_i is the time when task i is activated in solution S. A positive integer parameter Z works as a radius for the neighborhood. During the search, each task i may be activated between times $S_i - Z$ and $S_i + Z$. So, the larger the value of Z the larger the size of neighborhood and the longer the search might take. In practice, $Z = 4$ is large enough to find a better neighbor if it exists, but this parameter might need to be better calibrated according to the instance used. A time limit may be given to the local search procedure because, in hard instances, to look into the whole neighborhood might take a very long computational time.

$$y_{it} = 0 \quad \forall i = 1, ..., n \quad \forall t = 1, ..., S_i - Z - 1 \tag{15}$$

$$y_{it} = 1 \quad \forall i = 1, ..., n \quad \forall t = S_i + Z, ..., H \tag{16}$$

This first hybrid heuristic is called EA_ages+LS_cplex because the LS_cplex is used as an improving tool when a new best solution is found or a large exploration is done. At these times, Eq. (15) and Eq. (16) are added to the formulation F2, according to the values of Z and the solution S passed to LS_cplex. If LS_cplex finds a better neighbor, the algorithm not only records the solution value but also makes this neighbor an alive individual which is inserted into the current population. By this way, the individual can disseminate its genes through future recombinations, contributing to the composition of better future generations. The conversion from a CPLEX feasible solution back to an individual is done by giving higher priorities to tasks activated earlier in the CPLEX solution. It can be dome by multiplying $H - S'_i$ by K where S'_i is the time when task

i was activated in the neighbor S' and K is a positive real. Several values for K were tested, but no significant difference could be perceived among these values, at the end of the algorithm execution.

The second hybrid heuristic proposed in this paper is called EA_ages+CPLEX. It consists of a simultaneously cooperative execution of both EA_ages and CPLEX. EA_ages creates the initial population and calls CPLEX giving to it two information: the best schedule found so far and a list with one positive value for each binary variable of the formulation F2. These values mean the branching order which CPLEX must follow when choosing a binary variable to branch. According to the CPLEX documentation, this kind of list is used when the branch-and-cut algorithm has to choose a binary variable to fix its value if there are two or more variables with fractional values. So, the already fixed variables are skipped during the list traversal. The values are computed according to the time unit related. A variable related to the first time unit must be fixed earlier than a variable from the last time unit. Eq. (17) shows how the values were computed. The higher the value the earlier a variable must be fixed during the branch-and-cut algorithm.

$$Value(y_{it}) = H - t + 1 \quad \forall i = 1, ..., n \quad \forall t = 1, ..., H \tag{17}$$

Giving the CPLEX such list allows it to fix the variables in a more intelligent way because every earlier time unit has been fully done. Obviously, a better and faster sequence of variables branching might exists, but we expect that the proposed list (sequence of variables branching) works well in a general way.

After the CPLEX initialization, both EA_ages and CPLEX are running. They exchange information every time one of them finds a new best solution. EA_ages always give to CPLEX the new best schedule generated by it as the CPLEX initialization call did. This schedule is utilized by CPLEX to improve its primal bound. When a primal bound is improved in a branch-and-cut algorithm, several nodes may be removed from the tree. Thus, a better primal bound helps the optimization process by reducing the number of nodes and the memory utilization. If CPLEX finds a better primal bound by its own optimization process, it gives the EA_ages this information as the LS_cplex does, i.e., converting the CPLEX solution into a priority list that is inserted into the current EA_ages population. The stopping criterion of EA_ages+CPLEX is the end of the optimization process. When CPLEX proves the optimality of a certain solution it finishes the EA_ages and itself. Therefore, EA_ages+CPLEX may be considered an exact method while the EA_ages+LS_cplex is a heuristic method. Of course, when we give EA_ages a time limit it will behave as a heuristic method because no optimality might be ensured.

4 Computational Results

All tests were made with an Intel Quad-core Q9550 with 8 GB of RAM, running Linux (Ubuntu 8.10) an CPLEX 11.2 (parallel version up to 4 threads). Instances have been retrieved from LABIC project (`http://www.ic.uff.br/~labic`) and all have non-trivial solutions. Graphs are composed of 10% of tasks without precedence, other tasks have up to 5 predecessors. Costs and profits are randomly chosen in the range [1;50] and [1;10], respectively, in order to produce interesting instances. Higher costs would make only few tasks to be activated; lower values would provide very easy instances

Table 1 Primal and dual bounds obtained by mathematical formulations

	Primal bound		Dual bound	
Inst.	F1	F2	F1	F2
50a	47	47	47.0	47.0
100a	304	304	304.0	304.0
150a	576	576	576.0	576.0
200a	636	636	636.0	636.0
250a	1030	1030	1030.0	1030.0
300a	2073	2073	2073.0	2073.0
350a	2571	2571	2585.3	**2571.0**
400a	7265	**7271**	7304.9	**7271.0**
450a	9633	**9705**	10061.6	**9705.0**
500a	14222	**14336**	15925.2	**14336.0**
550a	-	**11087**	15401.4	**11136.5**
600a	-	**10157**	14968.1	**10199.6**
650a	-	**16332**	21800.2	**16355.8**
700a	-	**31820**	36406.8	**31836.2**
750a	-	**36216**	42069.0	**36296.9**
800a	-	**38351**	45307.0	**38729.8**
850a	45240	**46840**	53206.7	**46898.0**
900a	-	**38733**	47904.1	**38834.8**
950a	-	**67903**	75063.0	**67927.4**
1000a	-	**71864**	79287.9	**71911.8**

where all tasks could be activated. Planning horizon size was defined as square root of n (number of tasks) and the initial amount of resources Q_0 is enough to activate at least one task without precedence. P_0 is always equal to zero. These instances have been used by all papers about DRCPSP. Instances from other project scheduling problems are not suitable for the DRCPSP because they do not have profit values. Introducing artificial values in those instances and using them to compare the proposed formulation are not the objective of this paper.

Table 1 shows the primal and dual bounds found by CPLEX itself for each instance. A time limit of 50000 seconds was given to each execution. The best generated bounds (between the two formulations) are in bold. In small instances both formulations could find an optimal solution and prove its optimality. However, as the instances size increases, the gap between the bounds provided by F2 becomes much smaller than results obtained by F1. In many instances, even a single feasible solution was not generated by F1. The inequalities of the second formulation are better exploited during the simplex algorithm because to find a valid basis is easier when the constraints are simpler. Moreover, the coefficient matrix of F2 is much more sparse than F1. The better performance of F2 is also shown in Table 2. In this table, the times need to prove the optimality of a solution are presented. Again, F2 had better results taking less than half of the time spent by F1. Unfortunately, in instances with more than 500 tasks none of the formulation could prove the optimality of any solutions. The linear relaxation values provided by F2 are closer to primal bound than the values provided by F1.

Table 3 shows a comparison among the formulations and the best known heuristic for DRCPSP. A time limit was given to each execution. It was computed as the time needed by F2 to finish. In hard instances, this time was replaced by 7200 seconds. The heuristic results are the average of 30 independent executions. Even the formulation F2 being considered an exact method, it generated better results than the heuristic method did in a difficult situation where a very small amount of time was given. For most of the optimization problems, the limit of two hours is very tight in order to

Table 2 Time spent (sec) by each formulation - the time limit is 50000 seconds for each execution

Inst.	F1	F2
50a	0.0	0.0
100a	0.2	**0.1**
150a	0.5	**0.3**
200a	12.3	**6.5**
250a	88.2	**25.3**
300a	5674.3	**196.3**
350a	50000.0	**211.3**
400a	50000.0	**467.8**
450a	50000.0	**9843.2**
500a	50000.0	**6688.7**
550a-1000a	50000.0	50000.0

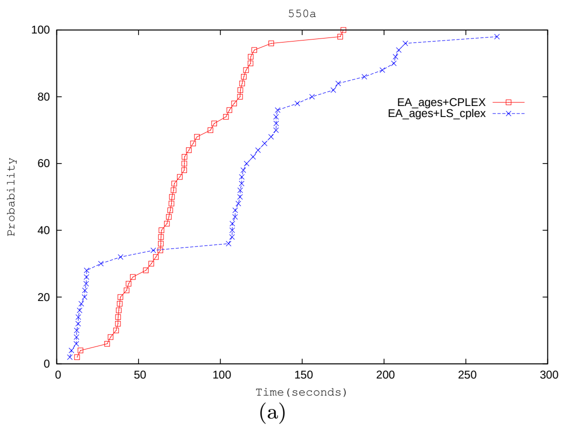
Table 3 Comparison among formulations and EA_ages parallel (heuristic approach) - the limits are the times F2 took to prove the solution optimality or 7200 second

Inst.	Limit(sec)	F1	F2	EA_ages Paral.
200a	6.4	602	636	636.0
300a	155.4	2052	**2073**	2002.6
400a	440.7	7167	**7271**	7062.1
500a	896.8	-	**14436**	13933.1
600a	7200.0	-	**10157**	9922.9
700a	7200.0	-	**31820**	30408.2
800a	7200.0	-	-	**37324.3**
900a	7200.0	-	**38683**	37045.5
1000a	7200.0	-	**71851**	69602.2

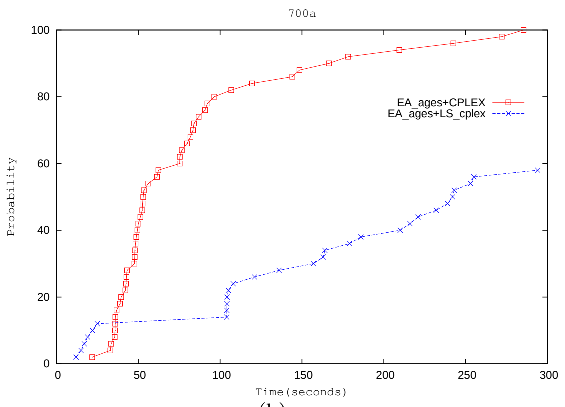
prove the optimality of a solution. This statement can be seen in instance 800a, where even a single feasible solution could not be found by the formulations. Comparing both formulations, we can see that F2 was again much better than F1. Thereby, the formulation F2 can be considered empirically better than the formulation F1 and will be used in hybrid heuristics instead of F1. The EA_ages parallel results, not as good as F2 results, are largely because the heuristic nature of the algorithm. Even giving more hours to EA_ages, it could not keep improving the solution. This behavior may be considered a kind of "natural limitation" of that algorithm. In other words, these results represents the expected solution value if given a large amount of time to EA_ages. In fact, a experiment was done giving about 20 hours to the parallel version and the obtained results were quite similar to those shown in Table 3. Unlike heuristic methods, the formulation executed by CPLEX can keep improving the solution until reaching an optimal one, if enough time is given.

Table 4 shows the results of a test very similar to that presented by Table 3. Now, parallel EA_ages was compared to the hybrid versions EA_ages+CPLEX and EA_ages+LS_cplex. The same time limit criterion was applied. The hybrid results are average of 30 executions. EA_ages+CPLEX had slightly better results than EA_ages+LS_cplex in almost all instances. Results from both hybrid versions exceeded the heuristic ones in all tested instances, except 200a. As said before, the hybrid heuristic may be considered heuristic version when a time limit is given because they will not be allowed to run up to the optimality state.

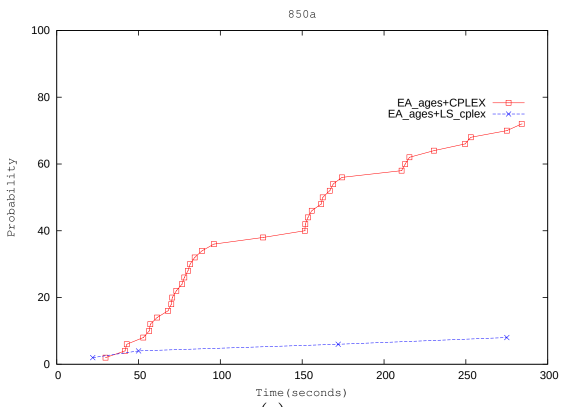
Fig. 3 present a probabilistic analysis. To plot the empirical distribution, we fixed a solution target and run each algorithm 100 times, recording the time when a solution



(a)



(b)



(c)

Fig. 3 Probabilistic analysis of EA_ages+CPLEX and EA_ages+LS_cplex versions - the targets are the average results of EA_ages after 300 seconds

Table 4 Results from hybrid heuristics compared to EA_ages parallel version - the limits are the times F2 took to prove the solution optimality or 7200 second - "Improv." column shows how much a hybrid heuristic was better than "EA_ages Paral."

Inst.	Time Limit(sec)	EA_ages Paral.	+LS_cplex	+CPLEX	Improv.(%)
200a	6.4	**636.0**	**636.0**	**636.0**	0.0
300a	155.4	2002.6	2038.2	**2072.5**	3.5
400a	440.7	7062.1	7217.2	**7271.0**	3.0
500a	896.8	13933.1	14319.7	**14335.9**	2.9
600a	7200.0	9922.9	10107.0	**10157.6**	2.4
700a	7200.0	30408.2	31819.6	**31823.9**	4.7
800a	7200.0	37324.3	38626.1	**38637.0**	3.5
900a	7200.0	37045.5	38766.6	**38768.6**	4.7
1000a	7200.0	69602.2	**71855.8**	71843.0	3.2

Table 5 Time spent(sec) and best solution found by EA_ages+CPLEX and CPLEX (two configurations)

	Total time(sec)			Solution		
Inst.	DS0	BB4	Hybrid	DS0	BB4	Hybrid
300a	196.3	**178.9**	2460.4	2073	2073	2073.0
350a	211.3	408.6	**155.0**	2571	2571	2571.0
400a	467.8	1609.2	**163.7**	7271	7271	7271.0
450a	9843.2	7570.0	**1143.7**	9705	9705	9705.0
500a	6688.7	7228.1	**822.6**	14336	14336	14336.0
550a	25000.0	25000.0	25000.0	**11081**	11054	11054.7
600a	25000.0	25000.0	25000.0	10157	10155	**10159.2**
650a	25000.0	25000.0	25000.0	**16332**	16308	16331.3
700a	25000.0	25000.0	**15061.2**	31820	31818	**31826.0**
750a	25000.0	25000.0	25000.0	36216	36220	**36225.7**
800a	25000.0	25000.0	25000.0	-	-	**38641.1**
850a	25000.0	25000.0	25000.0	46835	-	**46837.2**
900a	25000.0	25000.0	25000.0	38723	38650	**38787.9**
950a	25000.0	25000.0	25000.0	67902	67854	**67909.0**
1000a	25000.0	25000.0	25000.0	**71851**	71844	71838.6

with value greater than or equal to the target is found. For each algorithm, we associated a probability $p_i = i/100$ with the i-the sorted running time t_i and plotted the points $z_i = (p_i, t_i)$ for $i = 1, ..., 100$. This kind of plot is called time-to-target [8] and is used to show the empirical behavior of heuristic methods. The target chosen was the EA_ages average results presented in Tables 3 and 4. A time limit of 300 seconds was also given. If an execution does not reach the target, its results are not plotted.

The results of EA_ages+LS_cplex were not as good as EA_ages+CPLEX results. The problem is in the neighborhood traversal. Even when the best neighbor is found, the traversal does not finishes until it looks into the whole neighborhood. LS_cplex does not have a mechanism to know that there is no better solution in the remainder of the neighborhood. This waste of time made the algorithm to improve the best solution only a few times. In the easiest of the three instances tested (instance 550a - Fig. 3(a)) the behavior of both hybrid heuristic was very similar, but in harder instances the small amount of time was crucial to determine the performance of the methods. EA_ages+LS_cplex cannot be considered an ineffective algorithm. It indeed obtained results better than EA_ages heuristic, which is the best known algorithm so far. In the last test, EA_ages+CPLEX will be directly compared to CPLEX by itself. Tables 5 and 6 present the results of these tests.

Table 6 Dual bounds and best solution times from EA_ages+CPLEX and CPLEX (two configurations) - a dash means that the dual bound is equal to the solution found

	Dual bound			Best solution time(sec)		
Inst.	DS0	BB4	Hybrid	DS0	BB4	Hybrid
300a	-	-	-	**155.3**	176.3	1266.8
350a	-	-	-	**70.1**	191.0	104.0
400a	-	-	-	440.6	1209.7	**120.4**
450a	-	-	-	9693.3	7544.5	**1113.9**
500a	-	-	-	896.8	851.2	**466.1**
550a	**11140.6**	11152.9	11162.2	9221.6	22826.2	**6802.0**
600a	10205.1	10239.8	**10197.9**	**4234.9**	18181.5	8845.4
650a	16363.4	16446.6	**16351.1**	**5878.1**	24931.1	14068.8
700a	31843.8	31854.0	**-**	**6966.7**	19862.4	14541.9
750a	36302.0	36316.9	**36291.1**	23388.6	20855.4	**19751.1**
800a	38736.3	38757.8	**38686.0**	25000.0	25000.0	**12336.2**
850a	**46912.9**	46968.6	46920.5	**21658.4**	25000.0	23233.0
900a	38837.0	38845.1	**38823.9**	**14641.6**	24919.5	15187.5
950a	67931.1	67943.7	**67916.8**	24982.8	17380.7	**14811.1**
1000a	**71914.1**	71920.1	71930.2	**4326.4**	24372.2	18270.8

Throughout all tests, a CPLEX parameter was adjusted in order to improve the hybrid results. This parameter is called "MIP Emphasis" by the CPLEX documentation. The objective of the emphasis is to intensify the search for an optimal solution in some way. The default is 0 which means no specific emphasis. Hybrid version obtained better results when the parameter was adjusted to value 4 which means an intensification in searching for "hidden feasible solution". Unfortunately, the documentation does not give more details about how it is done. We believe that when a node of the branch-and-cut is analyzed, the CPLEX makes a "more accurate guess" about the variable values that might produce a feasible solution.

Other trouble we had when implementing the EA_ages+CPLEX algorithm is about the CPLEX execution mode. According to the documentation, CPLEX has two modes: dynamic search and traditional branch-and-bound. Again, no detailed information is given about the difference between these two modes. EA_ages+CPLEX was implemented with traditional branch-and-bound mode because the log of the CPLEX tells that the utilization of any callback automatically changes the default mode from "dynamic search" to "traditional branch-and-bound".

In any case, our hybrid version was compared to both modes. A first configuration was defined using the CPLEX defaults like in the first experiment (Table 1). No parameter except the time limit was changed. This configuration will be denoted by DS0. A second configuration is more similar to our hybrid heuristic. It uses "traditional branch-and-bound" and emphasis 4. This configuration will be denoted by BB4. The hybrid EA_ages+CPLEX was executed 10 times and the average results were taken. A time limit of 25000 seconds was given to each algorithm.

Table 5 presents the total time taken by each version and the best solution generated by them. DS0 produced better solution than BB4 did. Due to the lack of an appropriate documentation, to affirm which algorithm component was the responsible for the results is not possible. EA_ages+CPLEX performed very well compared to CPLEX by itself; both solution and total time averages of the hybrid heuristic were better. Instance 700a is a very interesting case. This instance is a hard one - none of the formulations found an optimal solution even with 50000 seconds (Table 1). However, the hybrid heuristic found an optimal solution (with value 31826) and proved it. Moreover, the algorithm spent about only 15000 seconds to do this. Comparing to the first test, the algorithm took less than one third of the time spent by CPLEX by itself to produce a better solution. CPLEX is a software very well recognized commercial and academically, but it can be improved a lot by inserting some specific problem knowledge into it. The solutions (primal bounds) and the branching order are examples of knowledge that CPLEX can make good use of.

Table 6 exhibits additional results: the dual bounds and the time when the final solution was found. These last results are not so conclusive because a solution can be found very quickly but it may also have a poor quality. In any case, the hybrid times were lower in some instances and higher in others.

EA_ages+CLPEX improved the dual bounds of many instances when compared to DS0. The main responsible for these results is the list of branching order. The branch-and-bound nodes were traversed in a more intelligent way making the dual bound decrease a little faster. The dual bounds from the hybrid heuristic are lower even than the bounds presented in Table 1, where CPLEX run for 50000 seconds.

5 Conclusion

This paper deals with a scheduling problem which uses an uncommon kind of resource: the Dynamic Resources. The Dynamic Resource-Constrained Project Scheduling Problem (DRCPSP) has application in industrial and commercial environments.

A new mathematical formulation (F2) is proposed for the problem. The binary variables of F2 have a different meaning from the previous formulation F1. Variables now provide information about the past of the task they are related to. To know if a task is activated becomes easier. Empirical tests shown that F2 can produce better solution in a smaller amount of time.

Formulation F2 was applied into a hybrid scheme with the best known heuristic (EA_ages) for DRCPSP. Two hybrid heuristics were proposed in this way. The first one (EA_ages+LS_cplex) used the formulation (and the CPLEX optimizer) as a local search tool. At some times, the search was executed and the best neighbor found might be inserted into the current population of the evolutionary algorithm. The second hybrid heuristic (EA_ages+CPLEX) executed both EA_ages and CPLEX simultaneously. They exchange information at the beginning of the running and when one of them finds a new best solution.

Some tests demonstrated the potential of the second hybrid heuristic in finding better solution and producing lower dual bounds. This algorithm is slightly better than the EA_ages+LS_CPLEX. Some comparisons with the second hybrid heuristic and CPLEX alone were done. CPLEX results were not as good as the hybrid ones. Even if we let the CPLEX run twice as long the hybrid heuristic does, the CPLEX results were worse. EA_ages+CPLEX presented the best primal and dual bounds for DRCPSP and it can be considered the best heuristic for this problem.

References

1. P.-H. Chen and S. M. Shahandashti. Hybrid of genetic algorithm and simulated annealing for multiple project scheduling with multiple resource constraints. *Autom. in Constr.*, 18:434–443, 2009.

2. N. Damak, B.Jarboui, P.Siarry, and T.Loukil. Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Comput. & Oper. Res.*, 36:2653–2659, 2009.
3. M. Fischetti, and A. Lodi. Local Branching. *Mat. Prog.*, 98:23–47, 2003.
4. M. Fischetti, and A. Lodi. Repairing MIP infeasibility through local branching. *Comp. & Oper. Res.*, 35:1436–1445, 2008.
5. J. Gonçalves, J. Mendes, and M. Resende. A random key based genetic algorithm for the resource constrained project scheduling problems. *Int. J. of Prod. Res.*, 36:92–109, 2009.
6. J. Homberger. A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *Int. Trans. in Oper. Res.*, 14:565–589, 2007.
7. J. Puchinger and G. R. Raidl. An Evolutionary Algorithm for Column Generation in Integer Programming: an Effective. In *Parallel Problem Solving From Nature - PPSN VIII, Volume 3242 of LNCS*, 642–651, 2004.
8. C.C. Ribeiro, and I. Rosseti. Efficient Parallel Cooperative Implementations of GRASP Heuristics. *Paral. Comp.*, 33:21–35, 2007.
9. Y. Semet and M. Schoenauer. An efficient memetic, permutation-based evolutionary algorithm for real-world train timetabling. In *Congress on Evolutionary Computation*, 2005. IEEE Press.
10. A. R. V. Silva and L. S. Ochi. A dynamic resource constrained task scheduling problem. In *Proc. of Lat.-Ibero-Am. Congr. on Oper. Res.(CLAIO)*, Montevideo (Uruguay), November 2006.
11. A. R. V. Silva and L. S. Ochi. A hybrid evolutionary algorithm for the dynamic resource constrained task scheduling problem. In *Proc. of the Int. Workshop on Nat. Inspired Distributed Comput. (NIDISC'07)*, LongBeach (EUA), March 2007.
12. A. R. V. Silva, L. S. Ochi, and H. G. Santos. New effective algorithm for dynamic resource constrained project scheduling problem. In *Proc. of Int. Conf. on Eng. Optim. (ENGOPT)*, Rio de Janeiro (Brazil), June 2008.
13. A. R. V. Silva and L. S. Ochi. New sequential and parallel algorithm for dynamic resource constrained project scheduling problem. In *Proc. of Int. Workshop on Nat. Inspired Distributed Comput. (NIDISC'09)*, Rome (Italy), May 2009.
14. J. Till and G. Sand and M. Urselmann and S. Engell. A hybrid evolutionary algorithm for solving two-stage stochastic integer programs in chemical batch scheduling *Computers & Chemical Engineering*, 5:630-647, 2007.