Creating Hybrid Solutions for Inventory Management Problems (extended abstract)

Robert Rodošek
IC-Parc, Imperial College
London SW7 2AZ, England
E-mail: r.rodosek@doc.ic.ac.uk

Tibor Kökény, Yves Caseau, Claude Le Pape
Bouygues, Challenger
1 av Eugene Freyssinet
78061 St-Quentin-Yvelines Cedex, France
Email: {tkokeny, ycs, clp}@challenger.bouygues.fr

Abstract

This paper presents an ongoing research in developing hybrid algorithms suitable for complex optimisation problems in CHIC-2 project (Esprit Project 22165). We present the modelling and solving of the Bouygues' application by two project partners: IC-Parc and Bouygues. The Bouygues' application is an inventory management problem which consists of two subproblems: a resource allocation subproblem and a maintenance scheduling subproblem. We demonstrate hybrid approaches which improve significantly the current results with respect to cost, quality, and integration.

1 The Bouygues' Application

The CHIC-2 project tackles several applications, each supplied by a major end user in the project. The applications are of strategic business importance, illustrate the blending of combinatorial optimisation and domain-dependent constraints. The aim of the project is to develop hybrid algorithms suitable for industrial optimisation problems, to provide a methodology to support efficient exploitation of the algorithms, to test and validate the algorithms on real problems, and to build a platform which can support their efficient and flexible execution.

The first principal objective is to design and deliver a range of hybrid algorithms for complex optimisation problems. These are based on the integration of three different approaches, i.e. constraint logic programming, mathematical programming, and stochastic repair techniques. The need for such algorithms in addressing optimisation problems arises from the fact that different techniques are suited to solving different parts or aspects of the problems.

The Bouygues' application is an inventory management problem which consists of two subproblems: a resource allocation subproblem and a maintenance scheduling subproblem. The resource allocation subproblem deals with the allocation of the equipment to requests and consists in determining which requests have to be subcontracted, which items are allocated to those requests that are not subcontracted, and when to buy new items. Substitutability of items is also considered in this process: an item of type t may be replaced by an item of type t'; this may lead to a better solution if taking an item of type t' from the available stock is less expensive than subcontracting an item of type t. The scheduling subproblem deals with the work in the maintenance workshop and consists in determining at which time the workers maintain the items. Both subproblems do not use the same resources. The resource allocation part deals with items while the scheduling part considers workers in the workshop. The problems are connected by hard constraints that no item is used more than its maximal using time without maintenance.

We present the results of the Bouygues' application by two partners of the project: IC-Parc and Bouygues. Depending on resource types and user's demand, three simplifications of the inventory management problem have been investigated. A problem can be considered without maintenance (i.e. for prefabricated frameworks there is no need to maintain) and/or without purchase (i.e. for the considered period there is no budget for the purchase) and/or considering or not the substitutability issue. In real-life problems, predefined data is in general uncertain, especially when the time period under consideration is long. We assume that the time period is not too long, so that the data can be considered certain.

2 Different Approaches

2.1 Resolution without decomposition

The data model used by Bouygues is an object-oriented implementation in the CLAIRE language [1]. Concerning input data, order requests, resource types, and resources are defined as instances of the corresponding classes. The constraints are not explicitly represented. The verification is done during the solution search. The cost function is implemented by a cost procedure that computes the cost of the current state at each moment during the resolution.

The problem is solved without decomposition into subproblems. We use *constraint programming* with a *branch-and-bound* search. The principle of the resolution is the following: for each resource of each resource type, an order assignment is determined with maintenance scheduling. Step by step one resource is processed at the same time. The algorithm processes resources from the less specific to the most specific resource types. If a resource is scheduled (order assignment), there is no backtrack to this schedule. Maintenance and assignment constraints are satisfied for the schedule of each resource in accordance with the resources already scheduled.

2.2 Resolution with decomposition

The problem can be decomposed into two subproblems which represent a resource allocation and a maintenance of items. There are more reasons for this decomposition. First, this is a natural decomposition for the user. The subproblems represent combinatorial problems on different sets of resources. Second, the cost of a solution to the resource allocation subproblem represents a major part of the global cost. Third, the size of the problem may vary according to the type of resources but the search space is large already when we consider up to 100 requests, 50 items, and only a maintenance capacity of 2.

The resource allocation part (the core problem), can be regarded as an extension of a multi-dimensional knapsack problem. Several branch-and-bound algorithms with good average-case performance on knapsack problems have been presented in the literature [4]. Mixed integer programming is a good candidate for solving this problem. On the other hand, the scheduling part is usually better to solve with constraint reasoning. Finite domain solvers have been successfully used on scheduling applications [6].

The decomposition applied by Bouygues and IC-Parc consists in resolving the core problem (to optimality if possible) and then trying to place maintenance times. However the resolution of the core problem and the maintenance scheduling is completely different in both approaches.

The main idea in the Bouygues' approach is to build a sort of set covering problem for each core problem and resolve it in a heuristic manner with local optimisation. The resolution of the core problem will generate several solutions. When a solution is found for the core problem, we try to place maintenance times and save the globally best solution. The strategy for buying can be implemented in a similar manner to the maintenance. We use *constraint programming* and the approach is based on a *truncated branch-and-bound* algorithm coupled with a *local optimisation procedure*. The branching is done on the choice of the request and the number of resources to rent for this request is the minimal value of the intervals covered by this request. After performing a large number of measures about the position of the optimal solution in the search tree, we have implemented a non-symmetrical search algorithm where the number of failures for left nodes is bounded. This search is coupled with a local optimisation that is applied for each solution obtained during the search. The local optimisation moves shift previously scheduled maintenance periods to increase the efficiency of the algorithm.

The IC-Parc's algorithm is a hybridisation of *constraint logic programming* and *mixed integer programming (MIP)*. The problem is solved by combining two solvers: the finite domain solver in ECLiPSe [3] and the MIP solver in CPLEX [2]. Constraint logic programming is used as a modelling language for both solvers. The problem is defined in ECLiPSe and the program is unfolded into the conjunction of linear constraints which represent a mathematical programming model [5]. Since the ECLiPSe platform uses CPLEX as a MIP solver, the linear constraints are automatically considered by CPLEX which solves the problem and returns the optimal solution to ECLiPSe. The principle of the resolution is the following: the MIP solver derives the optimal solution to the core problem with purchase and substitutability, and the finite domain solver extends this solution to the global solution by deriving also the maintenance of items. If such an extension does not exist, the MIP solver is performed again on a modified core problem containing requests with longer time durations. This procedure is repeated until a solution of the whole problem is derived.

2.3 Empirical results

The results of the proposed approaches show that the combination of different solvers offers genuine practical advantages over particular solvers.

In order to experiment and test different algorithms developed by the partners, a set of different problem instances have been defined. These benchmarks are different in complexity and size. We randomly generated a set of problem instances which can be considered without purchase and/or without substitutability and/or without maintenance.

Each of the three approaches described above appears to perform better on some but not all of the instances. The Bouygues' algorithm without decomposition considers each resource item one after other. In practice, it works well when maintenance is important and the number of requested items per order is small. Its main force is the use of a nearly full maintenance scheduling after each choice in the search tree. The main disadvantage is the computing time that varies from 10 second to 15 minutes.

Considering the algorithms with decomposition, the core problem instances are quite large and a heuristic approach in the Bouygues' algorithm such as the truncated branch-and-bound cannot produce the optimal solutions that are found by the linear programming approach of IC-Parc. However the local optimisation step does a good job at finding a near optimal solution. Usually the solution found at the branch-and-bound node is still 3-5% from the optimal, but the final solution is within a 0-0.7% range. However, it takes some time to tune the bounded search and the local optimisation though. The algorithm has been performed on problem instances without purchase. The next step is to set some boundaries for buying and to tune the search for the whole problem.

For the core problem with purchase and substitutability the method proposed by IC-Parc computes the optimal solution (in most cases, without branching). The empirical results demonstrated that the integration of the mathematical programming solver on the resource allocation subproblem and the finite domain solver on the scheduling subproblem successfully solves the whole problem in reasonable time. The main advantage of this approach is the robustness of the hybrid algorithm. On some instances, however, the obtained global solution is still a few percents above the best known lower bound. Further research aims at developing an improved hybrid algorithm incorporating the linear programming and the finite domain solver in a more intricate way.

References

[1] Y. Caseau and F. Laburthe. CLAIRE: A Parametric Tool to Generate C++ Code for Problem Solving. *Working Paper, Bouygues, Direction Scientifique*, 1996.

[2] CPLEX. Using the CPLEX Callable Library, Version 3.0. *CPLEX Optimization, Inc.*, 1994.

[3] ECLiPSe Version 3.5.2 User Manual. *In Technical Report, IC-Parc, Imperial College, London*, 1996.

[4] S. Martello and P. Toth. Knapsack Problems. *John Wiley&Sons*, 1990.

[5] R. Rodošek, M. G. Wallace, M. T. Hajian. A New Approach to Integrate Mixed Integer Programming with CLP. *In Proceedings of the CP96 Workshop on Constraint Programming Applications: An Inventory and Taxonomy*, pp. 45-54, Cambridge, Massachusetts, 1996.

[6] P. van Hentenryck. Constraint Satisfaction in Logic Programming. *MIT Press*, 1989.