Improving Wedelin’s Heuristic with Sensitivity Analysis

Sara Maqrot\(^1\), Simon de Givry\(^1\), Marc Tchamitchian\(^2\), Gauthier Quesnel\(^1\)

\(^1\) UR 875 MIAT, Université de Toulouse, INRA, Castanet-Tolosan, France
\(^2\) UR 767 Ecodéveloppement, INRA, Avignon, France

\{firstname.lastname\}@inra.fr

Keywords: integer programming, heuristics, Lagrangian relaxation, sensitivity analysis

Abstract

Heuristic is one of the important techniques designed to find quickly good feasible solutions for hard integer programs. Most heuristics depend on a solution of the relaxed linear program. Another approach, Lagrangian relaxation offers a number of important advantages over linear programming [4], namely it is extremely fast for solving large problems. One of the Lagrangian based heuristics is Wedelin’s heuristic [10], which works for the limited class of 0-1 integer programs. It is designed for problems with a specific form \((\min_{x \in \{0,1\}^n} \{c^T x \mid Ax = b\})\) with coefficients of \(A\) in \(\{0,1\}\) and \(b\) integral) which prevents its applicability on many problems.

Thus to tackle problems with more general structures, [1] presents a generalized Wedelin’s heuristic for integer programming. The performance of this method depends crucially on the choice of its numerous parameters (number of iterations, degree of approximation, history of the preference matrix, and others). To adjust these parameters and learn which ones have important influence on whether a solution is found and its quality, we conduct sensitivity analysis combined with a metaheuristic. We choose the Morris method [8] to select parameters providing a feasible/good solutions on a family of instances, and the genetic optimization using derivatives (genoud) method [7] to find the best solution in limited time for a specific instance. The Morris method consists in discretizing the input space for each parameter, then performing a given number of One At a Time random designs (each input parameter is varied while fixing the others). The repetition of these steps allows the estimation of elementary effects for each parameter, and consequently the sensitivity indices [5]. genoud combines an evolutionary algorithm method with a derivative based (quasi-Newton) method to solve difficult optimization problems. These difficulties often arise when the objective function is a nonlinear function of the continuous parameters and not globally concave having multiple local optima [7].

We have implemented a C++ parallel version of Wedelin’s heuristic based on [1]. The solver called baryonyx\(^1\) has two execution modes: solver mode and optimizer mode. In the solver mode, baryonyx runs once trying to satisfy all the constraints. In the optimizer mode, it runs in parallel according to the number of processors, and tries to satisfy all the constraints and to optimize the solution at each run, reporting the best solution found for all runs when it reaches its time limit. Concerning parameters, Morris and genoud methods are implemented in R packages. We use Morris package to find useful parameters. Once found, we fix other parameters and we let genoud adjusts the useful ones in order to get the best solution within a given time limit. We compare baryonyx with exact solver IBM ILOG cplex and with two local search methods: a 4-flip neighborhood local search algorithm [9] and an hybrid mathematical programming solver LocalSolver [2] which is a simulated annealing based on ejection chain moves specialized for maintaining the feasibility of Boolean constraints and an efficient incremental evaluation using a directed acyclic graph. The following Tables show that baryonyx is competitive with the existing solvers on a Set Partitioning Problem (SPP) benchmark [3], but has difficulties on a Mixed Fruit-Vegetable Crop Allocation Problem (MFVCAP) [6].

\(^1\)https://github.com/quesnel/baryonyx
## References


