Scheduled Penalty Variable Neighborhood Search

E. Manni[1] and B.W. Thomas[2]

[1] Dipartimento di Ingegneria dell'Innovazione, Università del Salento, Lecce, Italy
emanuele.manni@unisalento.it
[2] Department of Management Sciences, University of Iowa, Iowa City, Iowa, USA
barrett-thomas@uiowa.edu

1 Introduction

For many $\mathcal{NP}$-hard combinatorial optimization problems, the existence of constraints complicates the implementation of a heuristic search procedure. Standard approaches for handling these constraints include (i) preserving feasibility in the search sequence of solutions through appropriately designed move operators and/or problem encodings, (ii) repairing infeasible solutions to obtain a corresponding feasible solution, and (iii) penalizing infeasibility to direct the search towards feasible solutions. In this talk, we extend the well known Variable Neighborhood Search algorithm (VNS) to include dynamic constraint penalization. We specifically focus on what are known as scheduled penalty methods and call the new algorithm scheduled-penalty VNS (spVNS). A scheduled penalty increases the level of penalization at each iteration of the search according to a pre-determined schedule. The iterative increase of the penalty has the effect of gradually driving the search from infeasible to feasible regions of the search space. In Section 2, we describe our solution approach, whereas in Section 3 we give an insight about the computational results we will present.

2 Algorithmic Design

We take as given the following combinatorial optimization problem:

$$\begin{aligned} \textbf{CP} \quad \text{minimize} \quad & v(x) = f(x) && (1)\\ \text{subject to} : \quad & g_i(x) \geq b_i; \quad i = 1, \ldots, m && (2)\\ & h_j(x) \geq d_j; \quad j = 1, \ldots, q && (3)\\ & x \in \mathcal{S}, && (4) \end{aligned}$$

where $\mathcal{S}$ is a finite set, and $f(\cdot)$, $g_i(\cdot)$, $h_j(\cdot)$ are real-valued functions on $\mathcal{S}$. To recover well-defined neighborhoods for our search, we relax constraints (2) with a specific penalty term. Specifically, for each $x \in \mathcal{S}$, let $p(x)$ be real-valued, nonnegative functions such that $p(x) = 0$ if and only if $g_i(x) \geq b_i$ for $i = 1, \ldots, m$. Let λ be a nonnegative, scalar penalty multiplier λ. Then, we formulate $\mathbf{RP}(\lambda)$, a relaxation of **CP**, as

$$\begin{aligned} \mathbf{RP}(\lambda) \quad \text{minimize} \quad & v'(x, \lambda) = f(x) + \lambda p(x) && (5)\\ \text{subject to} : \quad & h_j(x) \geq d_j; \quad j = 1, \ldots, q && (6)\\ & x \in \mathcal{S}. && (7) \end{aligned}$$

Our aim is to solve problem $\mathbf{RP}(\lambda)$. As our computational results will show, spVNS is capable of doing so in a way that finds high quality solutions that are also feasible to **CP**. The procedure, depicted in Algorithm 1, takes as input a function $v'(x, \lambda)$ that evaluates a solution x of the chosen problem instance for a given value of the penalty multiplier. We assume that $v'(x, \lambda)$ is obtained as a modification of the true objective v, accounting for the relaxation of particular problem constraints. The augmented objective $v'(x, \lambda)$ penalizes the violation of these relaxed constraints with the penalty parameter λ. The algorithm also requires an input parameter denoting the maximum number of penalty updates $i_{\max}$, and another one denoting the maximum neighborhood size $k_{\max}$.

The problem is initialized with a solution, not necessarily feasible, x. As with traditional VNS implementations (for discussion, see [1]), the solution is perturbed by a function Shake(x, k) that randomly selects x', a neighbor of x from neighborhood $\mathcal{N}_k(x)$. A local search procedure LocalSearch(x', λ) is then run with the intent of finding an improved solution x''. If x'' is improving,

Algorithm 1 Scheduled Penalty Variable Neighborhood Search

Input:
  Data for a problem instance including a function $v'(x, \lambda)$ that determines the value of a solution x with regard to a penalty λ for a measure of the violation of relaxed constraints
  A maximum number of penalty updates $i_{\max}$
  A maximum neighborhood size $k_{\max}$
Output: Solution, x
Initialization:
  Determine solution x
  $i = 1, k = 1, \lambda = 0$
while $i < i_{\max}$ **do**
    $x' \leftarrow \text{SHAKE}(x, k)$.
    $x'' \leftarrow \text{LOCALSEARCH}(x', \lambda)$
    if $v'(x, \lambda) > v'(x'', \lambda)$ **then**
      $x \leftarrow x''$
    else
      $k \leftarrow \text{UPDATEPERTURBATIONLEVEL}(k, k_{\max})$
    end if
    $\lambda \leftarrow \text{UPDATEPENALTY}(\lambda, i)$
    $i \leftarrow i + 1$
end while

then we update x with x''. Otherwise, we update the perturbation level (UPDATEPERTURBATIONLEVEL$(\cdot, \cdot)$ function) by letting $k = k + 1$. We note that the solutions are evaluated with respect to the objective $v'(\cdot, \lambda)$ throughout the algorithm. At the end of each improvement cycle, the penalty parameter λ is updated by the function UPDATEPENALTY(λ, i). As we focus on scheduled penalty methods, we assume that the update of λ depends only on the current iteration i and the current penalty λ.

3 Computational Results

To test the effectiveness of the proposed algorithm, we test our approach on well known benchmark instances of the traveling salesman problem with time windows and the orienteering problem with time windows. Our goal with these tests is two-fold. First, the results justify choices in algorithmic design. Second, through comparisons to state-of-the-art methods for the previously described test instances, the results demonstrate the quality of the proposed general heuristic framework. In addition to finding high quality solutions to the problems in the test sets, in some cases *best known solutions*, our computational experiments identify several considerations in implementing VNS with scheduled penalties:

1. the quality of the results is sensitive to the neighborhoods used in the shake phase,
2. high-quality results are achieved with much lower levels of perturbation than in standard VNS implementations,
3. while the perturbation levels are lower, scheduled penalty VNS often requires a large number of VNS iterations to reach high quality solutions. This result supports our implementation that separates the VNS iterations from the perturbation level, and
4. when using VND as the local-search function, the method performed well using neighborhoods well known in the literature.

References

1. Hansen, P., Mladenović, N., Brimberg, J., Moreno Pérez, J. Variable neighborhood search. In: Gendreau, M., Potvin, J.-Y., editors. Handbook of Metaheuristics. International Series in Operations Research & Management Science; Springer; 2010, p. 61–86.