Reactive variable neighborhood search

Thevenin, Simon; Zufferey, Nicolas

Reactive Variable Neighborhood Search

Simon Thevenin[1] and Nicolas Zufferey[2]
[1]HEC Montreal, Canada, simon.thevenin@hec.ca
[2]Geneva School of Economics and Management (GSEM) - University of Geneva, Switzerland, n.zufferey@unige.ch

Abstract
Recent works have shown that the variable neighborhood search (VNS) algorithm can be improved by using a reactive component. A component is reactive if its behavior changes dynamically depending on the information obtained from the current run, or depending on the instance to solve. This paper gives a review of these recent works, and proposes a new scheme to select reactively the move used in the shaking step of VNS. This new mechanism is particularly relevant to solve problems involving various types of decisions. For instance, an application of the resulting Reactive-VNS to a short-term production-planning problem is given. Experimental results show that Reactive-VNS outperforms the classical VNS approach on the latter problem.

Keywords: variable neighborhood search, reactive search, self-adaptive algorithm, guided search, hyper heuristics

1 Introduction

Variable Neighborhood Search (VNS) is an optimization method able to find high-quality solutions to complex and large problems in a short computation time. Good performances of VNS have been reported for different types of problems [5]. Recent works have shown that a reactive component can improve VNS. The reactive component is modified dynamically (during the search) depending on the instance to solve, or depending on the history of the search process. The resulting extension of VNS is called *Reactive Variable Neighborhood Search* (RVNS).

The contributions of this paper are three fold. First, a review of the works proposing to extend VNS with a reactive component is given (Section 2). Second, a new reactive mechanism is proposed (Section 3). This mechanism helps RVNS to escape from the attraction area of the best found solution so far. Finally, the resulting method is applied to a short-term supply chain problem (Section 4). Experiments show that RVNS outperforms VNS in terms of solution quality. A conclusion ends the paper.

2 Literature review

Combinatorial optimization methods aim at finding the solution with minimal cost among a finite set of feasible solutions (called solutions space). For large instances of NP-hard problems, finding the optimal solution in a reasonable amount of time is not possible. In such cases, a good-quality solution can be found in a reasonable amount of time using a meta/heuristic (e.g., greedy procedures, local search techniques, evolutionary algorithms). VNS belongs to the family of local search approaches. Two streamline local search methods are first discussed below (namely, descent and tabu search), followed by VNS and RVNS.

Local search algorithms are meta/heuristics, which navigate in the solution space from a solution to a neighbor solution. A neighbor s' of a solution s is a solution with a close structure to s, obtained by applying one or multiple perturbations (called moves) to s. The set of neighbors of a solution is called neighborhood. The descent algorithm is a simple local search that selects the best solution of the neighborhood at each iteration. However, every descent gets trapped in the first local optimum, as non-improving moves are forbidden. Tabu search (TS) overcomes this issue (i.e., it is able to escape from a local optimum) with a tabu list which prevents performing the reverse of recently performed moves. TS terminates when a given stopping criterion (e.g., a time limit) is reached.

Despite some escaping mechanisms, the outcome of TS (as many streamline local search procedures) depends on the single provided initial solution. Based on this observation, as long as a stopping condition is not met, VNS [5] successively applies a local search LS with different, well-chosen initial solutions. These initial solutions are generated during the so-called shaking step of VNS. This step requires the definition of multiple neighborhood structures $(N_{i_{min}}, \ldots, N_{i_{max}})$. The neighborhoods are sorted such that the neighborhood $N_i(s)$ contains solutions more similar to s (with respect to its overall structure) than $N_j(s)$ if $i < j$. Therefore i is an indicator of the distance between the solution in $N_i(s)$ and s. For instance, the solutions of $N_i(s)$ can be generated by sequentially applying i basic moves to s (e.g., a swap of two elements if a solution can be encoded as a list). In each iteration of VNS, the shaking step randomly generates a solution s_i in the neighborhood $N_i(s^\star)$ of the currently best found solution $s^\star$. The neighborhood distance i is set to its minimal value (i_{min}) at the beginning of the search, but also each time a new best solution is found (in order to intensify the search around this updated $s^\star$ solution). However,

if the solution returned by LS does not improve $s^\star$, i is increased in order to gradually explore zones of the search space away from $s^\star$. Though, a maximal distance value (i_{max}) is defined to avoid a random search (indeed, without such a restriction, generating a too distant neighbor solution corresponds to a restart of the method).

In this paper, RVNS denotes the extension of VNS where any component or parameter is adjusted during the search. Note that various names are used in the literature (e.g., adaptive VNS, guided VNS). The choice of the reactive element depends on the problem to solve. In [7], the random generation of a neighbor in the shaking phase is biased. Multiple ways to bias the shaking step are defined, and one is chosen reactively. Such an approach is efficient when the random selection of the neighbor leads to bad solutions. The authors of [2] propose to select the local search reactively among multiple options (each one using a different move). The reactive component in [3] and [6] is the neighborhood used in the shaking step.

Multiple strategies for controlling the behavior of the reactive component have been proposed. The most common is to use a learning mechanism [2, 3, 7]. In this case, the reactive component has multiple variants, and one variant is chosen at each iteration. A simple learning mechanism [3, 7] associates a score with each possible variant, and the probability to select a variant is proportional to its score. After each iteration, the score of the selected variant is increased if a new record is found. Advanced learning mechanisms can also be used to allow different reactions depending on the state of the search. For instance, in [2], reinforcement learning is used to learn a policy which defines the local search (reaction) to use next, given the previously used local search (state). Using a learning mechanism is relevant when the performance of the component variants depends on the instances to solve, or on the current search state.

Reactive strategies for VNS are not always based on learning. For instance, an approach to order the neighborhoods when no obvious ordering exists is proposed in [6]. First, the problem of finding the best neighbor in each neighborhood is solved by relaxing the integrality constraints. The solutions to these relaxed problems approximate the quality of the neighborhoods, and the neighborhoods are then ordered by decreasing quality. In [1, 4], the behavior of VNS is modified when the last neighborhood (i.e., when $i = i_{max}$) is reached. This is relevant when the new behavior diversifies the search, as reaching the largest value of i indicates that the method struggles to improve $s^\star$. In [4], a large move is applied to the current solution, whereas in [1], the objective function is reactively adjusted to drive the search away from local optima.

3 Reactive move selection

This section proposes a new scheme to select reactively the move used to generate a neighbor solution in the shaking step. Based on the information obtained from the last performed run of LS (i.e., the employed local search within VNS), the process selects a move allowing to escape from the attraction zone of the incumbent solution, simply denoted here as s.

In local search methods, the neighborhood is often generated with multiple basic moves. In fact, using multiple moves helps to connect the search space, as for any given pair of solutions s_1 and s_2, it must be possible to navigate from s_1 to s_2 by performing a succession of moves. Moreover, an increasing number of studies tackle problems by integrating decisions of different types (e.g., job selection and job scheduling). For such problems, it is natural to define one move for each type of decision. In a classical local search method, these multiple moves are used jointly to create a single neighborhood for each solution (e.g., job selection moves and job scheduling moves are used at the same time within the same neighborhood). In the proposed RVNS, multiple neighborhood structures are defined. Let N_i^k be the neighborhood with distance i generated with a move of type m_k. For instance, the solution in N_i^k can be generated by applying i moves of type m_k. In the proposed approach, the move used to generate the solution in the shaking step is selected reactively, and the distance i follows the usual VNS pattern (i.e., i is set to i_{min} when the solution obtained after local search is better than the incumbent solution, and i is increased otherwise).

When solving the short-term production-management problem defined in the next section, we observe that LS uses one type of move more frequently than the others. The most used move depends on the instance and on the initial solution of LS. Such a behavior indicates that the search is in a strong local optimum for one type of decision. If modifying decisions of a specific type leads to significant cost augmentations, LS naturally focuses on the modification of other types of decisions. Consequently, some decisions are never changed, and the search stays in a specific zone of the search space. To help escaping from this zone, the shaking step generates a solution with the least used move during the last LS run.

Algorithm 1 summarizes the proposed RVNS approach. It is relevant for problems involving various types of decision. A simple strategy to control the reactive component is proposed above. Although this control strategy performs well on the tested problems (see the next section), other strategies could be of course relevant for other problems. The principle is to define a strategy detecting that the search is trapped in a restricted area of the search space, and able to select a move which allows to escape from this region. For instance, a learning mechanism similar to the one proposed in [2] can be used to infer a policy returning the move to use depending on characteristics of the last LS runs.

Algorithm 1 Reactive Variable Neighborhood Search (RVNS)

Generate an initial solution s, set $i = i_{min}$, and set randomly k to one of the move type.

While no stopping criterion is met, **d**o

(1) *Shaking:* generate s' randomly in the neighborhood N_i^k.

(2) *Local search:* apply LS to s', using jointly all the move types in each iteration. Let s'' be the resulting solution, and set k to the least used move type in LS.

(3) *Move or not:* if s'' is better than s, set $s = s''$, and continue the search with $i = i_{min}$; otherwise, increase i, but if $i > i_{max}$, set $i = i_{max}$.

4 Application to short term supply chain management

This section presents the RVNS designed in [8] to solve a short-term supply-chain-management problem (P). The proposed reactive scheme is generalized in this paper, and this section aims at quantifying the benefit of RVNS for (P).

Given a set of orders, (P) consists in dimensioning production lots and scheduling them on parallel production lines. The decision maker is located at the plant level and the objective function includes four types of costs: (1) order rejection costs; (2) tardiness penalties (as the shops will be delivered later than expected); (3) sequence-dependent setup costs between the production lots; (4) raw material express delivery costs (to be paid to the suppliers to have the raw material earlier than initially planned). Specific features and constraints of (P) are not detailed here because they do not impact the design of RVNS (as well as the parameter tuning). In the next three paragraphs, considering (P), TS, VNS and RVNS are successively summarized.

The neighborhood of TS is generated with two types of moves: (m_1) insert an order o in a production lot (o is either a rejected order or an order planned in a different production lot); (m_2) modify the production day and the production line of a lot. After performing a move m_1, the order o gets a tabu status. Similarly, when moves m_2 are performed, all the orders of the lot get the tabu status. An order with a tabu status (or a lot containing a tabu order) cannot be moved for some iterations (parameter). To accelerate TS, only a random proportion (parameter) of the neighborhood is evaluated.

The local search in VNS is TS with a stopping criterion of 500 iterations. The neighborhood N_i used for the shaking step is generated by applying $h(i)$ (i.e., a function increasing with i) random moves in $m_1 \cup m_2$. In the shaking step, p (parameter) solutions are randomly generated in N_i, and the best one is the initial solution of TS. If the solution provided by TS does not improve the input solution, the distance i is increased.

RVNS enhances VNS by reactively selecting a single type of move m (m_1 or m_2) to generate the neighborhood in the shaking step. More precisely, $N_i^{m_1}$ (resp. $N_i^{m_2}$) is generated by performing again $h(i)$ moves of type m_1 (resp. m_2) on s. In the shaking step, RVNS chooses reactively to use $N_i^{m_1}$ or $N_i^{m_2}$. During each run of TS, the number of performed moves of each type (for both m_1 and m_2) is counted, and the least performed move is selected to generate the neighborhood in the next shaking step.

To measure the benefit of RVNS, the following comparisons are useful. First, comparing VNS (which uses TS as an intensification procedure) with TS allows capturing the added value of the shaking step (with the involved nested neighborhoods). Second, comparing RVNS with VNS allows quantifying the added value of the reactive scheme. The tests were performed on more than 50 instances, on a processor $IntelR\ XeonR\ CPU\ E5-2660\ 2.20GHz$, with a time limit of 20 minutes, and replicated with ten different seeds. Table 1 presents the results aggregated by number $|P|$ of products. For each group (i.e., line), Table 1 gives the average number $|O|$ of orders, the best cost

obtained by any of the tested method (averaged per group), and the average percentage gap with respect to the best cost (for each method). VNS outperforms TS with an average gap of 3.10% versus 5.32% for TS. The reactive move selection leads to an even better performance, as RVNS outperforms VNS with an average gap of 2.24%. In addition, RVNS obtains better solutions for each instance type.

$\|P\|$	$\|O\|$	Best cost	*TS*	*VNS*	*RVNS*
10	286	682,256.0	1.99	2.27	**1.27**
20	571	439,528.0	2.64	3.10	**1.94**
30	849	577,411.3	3.43	2.20	**1.22**
50	1426	805,332.8	5.27	3.62	**2.70**
100	2869	879,835.0	13.27	4.32	**4.09**
Average			5.32	3.10	**2.24**

Table 1: Comparison of TS, VNS and RVNS for (P).

5 Conclusion

This paper presents RVNS, a method obtained by adding a reactive mechanism to the standard VNS approach. It is relevant for combinatorial optimization problems involving different types of decisions. The reactive mechanism decides which optimization component to employ, either by using some learning techniques or some specific rules depending on the problem. We show that the discussed reactive scheme helps to better escape from local optima. Research on RVNS are relatively recent, but good performances are already reported. Investigating RVNS more deeply would be an interesting avenue of research. For instance, in most studies, a single component is reactive, and thus the use of multiple reactive elements could be considered. Except in [2], only simple learning mechanisms were proposed. Advanced learning methods could help to better predict which variant of a component would perform best in a particular state of the search.

References

[1] Bräysy, O. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4), 347–368, 2003.

[2] dos Santos, J. P. Q., de Melo, J. D., Neto, A. D. D., and Aloise, D. Reactive search strategies using reinforcement learning, local search algorithms and Variable Neighborhood Search. *Expert Systems with Applications*, 41, 4939–4949, 2014.

[3] Li, K., and Tian, H A two-level self-adaptive variable neighborhood search algorithm for the prize-collecting vehicle routing problem. *Applied Soft Computing*, 43, 469–479, 2016.

[4] Paraskevopoulos, D. C., Repoussis, P. P., Tarantilis, C. D., Ioannou, G., and Prastacos, G. P. A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. *Journal of Heuristics*, 14(5), 425–455, 2008.

[5] Hansen, P., and Mladenović, N., Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3), 449–467, 2001.

[6] Puchinger, J., and Raidl, G. R. Bringing order into the neighborhoods: relaxation guided variable neighborhood search. *Journal of Heuristics*, 14(5), 457–472, 2008.

[7] Stenger, A., Vigo, D., Enz, S., and Schwind, M. An adaptive variable neighborhood search algorithm for a vehicle routing problem arising in small package shipping. *Transportation Science*, 47(1), 64–80, 2013.

[8] Thevenin, S., Zufferey, N., and Glardon, R. Model and metaheuristics for a scheduling problem integrating procurement, sale and distribution decisions. *Annals of Operations Research*, 2017 (http://dx.doi.org/10.1007/s10479-017-2498-z)