A new hybrid method between VNS and SEA to improve results on the 0-1 multidimensional knapsack problem

Maroun Bercachi
bercachi@i3s.unice.fr
University of Nice-Sophia Antipolis - I3S Laboratory
2000 Route Des Lucioles - 06903 Sophia Antipolis - France

Abstract. Variable Neighborhood Search (VNS) has been successfully used to solve the 0-1 Multidimensional Knapsack Problem (MKP). In this paper, our contribution combines VNS with an evolutionary algorithm (EA). It leads to a new EA where the neighborhood of the mutation operator is iteratively changed in the same way as in VNS. The order in which the neighborhoods are visited in a VNS is crucial and difficult to fix at the beginning of the run. Our proposal will use a States based EA (SEA) to explore various orderings of the neighborhoods. The SEA is a parallel model which manipulates several EAs in a dynamic way. We tested this new approach on different instances of the MKP. Experimental tests show that our proposal favors the most appropriate order of the neighborhoods.

1 Introduction

The 0-1 Multidimensional Knapsack Problem (MKP) is a NP-hard problem which arises in several practical problems. It can be defined by the following mathematical statements: maximize $z = \sum_{j=1}^{n} p_j x_j$ subject to $\sum_{j=1}^{n} w_{ij} x_j \leq c_i$, $i = 1, \ldots, m$ where $x_j \in \{0, 1\}$, $j = 1, \ldots, n$. There are given $n$ items with profits $p_j > 0$ and $m$ resources with capacities $c_i > 0$. Each item $j$ consumes an amount $w_{ij} \geq 0$ from each resource $i$. The goal according to Eq. 1 is to select a subset of items with maximum total profit; chosen items must, however, not exceed resource capacities, see Eq. 2. The 0-1 decision variables $x_j$ indicate which items are selected. Some previous works (6) proposed a new variant of Variable Neighborhood Search (VNS) to solve this problem. In their method, the ordering of the neighborhoods is determined dynamically by solving relaxations of them. To our knowledge, the method currently yielding the best results on large instances, at least for commonly used benchmarks, has been proposed by Vasquez and Hao (8) and was recently refined by Vasquez and Vimont (4). It is a hybrid approach called V&V and based on tabu search. The second efficient algorithm so-called W&H that generates the best recent results for large instances is described in (5). The research presents several convergent algorithms that solve a series of sub-problems generated by exploiting some relaxations.

VNS is a recently proposed metaheuristic for solving combinatorial and global optimization problems (3). In order to reach local optima, a local search method
is repeatedly applied to these neighboring solutions. It explores increasingly larger neighborhoods of the current solution, jumping from this solution to a new one if and only if an improvement has been made. Else, it jumps to the next neighborhood. Working in this way, favorable characteristics of the current solution will be often kept and used to obtain promising solutions. Important issues in the VNS function need to be solved. In (3) the following significant question is raised when using a VNS: what is the best order in applying the neighborhoods? Usually, neighborhoods are sorted in order to increase time complexity of searching for the best moves. Sometimes, such an order might be hard to estimate or even be misleading. In general, a prespecified fixed order might not be the most adequate during all phases of the optimization process.

States based Evolutionary Algorithm (SEA) has been proposed in (1) as a hyperheuristic that operates at a high level of abstraction using several EAs simultaneously. The SEA is a generic and fast-to-implement method which should produce solutions of acceptable quality, based on a set of easy-to-implement low-level heuristics. More precisely, the SEA is a parallel algorithm with \( n \) states. An EA (GA, Genetic Program GP, Evolutionary Strategy ES, ...) is associated to each state. For each EA, the population is composed by all solutions having the same corresponding state. Therefore, each algorithm could have its proper parameter settings. The principle of the SEA is to select indirectly the best state according to the fitness values of the actual solutions applying a classical selection operator. Therefore, the SEA favors the most appropriate state while keeping the other states running in the pool. This is the main difference between the SEA and other hyperheuristics. By this way, the process will benefit from the characteristics of each state instead of simply choosing the best one and applying it solely. A changeState operator is repeatedly applied to solutions during the search. It could change their states according to a given ratio in order to preserve a sort of balance between the population size of each algorithm. Fitness values of solutions are kept the same while using this operator to prevent a misleading convergence towards one particular state. One global iteration of the SEA is the succession of the stochastic operators: selection, split, EA, changeState, merge and replacement. The algorithm stops once the total number of iterations or function evaluations is achieved. Different instantiations of the SEA have shown the utility of this parallel model for solving hard optimization problems (2).

We aim that our present study contributes in proposing a new heuristic to improve results previously obtained for the MKP. Since our proposal is not problem-dependent, it could also be used for solving other classes of optimization problems. The rest of this paper is divided into three sections. Section 2 presents our contributions. Section 3 displays the experiments and the results. In Section 4, we discuss the outcome and conclude about the effects of the new approach.

2 VN-EA and VN-SEA

Variable Neighborhood EA (VN-EA) uses an EA with a “family” of solutions while conserving the same principles of the basic VNS. VN-EA is not considered
as a hybrid approach between VNS and EA, rather it is a simple adaptation of the VNS to an EA. The local search method is an EA where the mutation operator is defined on $k$ different neighborhoods. This choice is based on the concept that an EA could be a good fast detector for local optima. Hence, this strategy can drastically accelerate the algorithm process. Consequently, VN-EA is expected to involve less computational power than a hybrid formulation. In VN-EA, the concept of neighborhood is extended to gain a more general construct. It will be considered as a probabilistic neighborhood. In this context, the neighborhood structures in VN-EA will be composed by a set of consecutive gene mutation ratios. In VN-EA, the neighborhood of the mutation operator is iteratively changed in the same way as in VNS. Our first intuition is based on the idea that in EAs, mutation serves the crucial role of replacing the gene values lost from the population during the selection process so that they can be tried in the new context. So, VN-EA starts by initializing a population, defining a set of neighborhood structures and fixing their specific ordering. Next, a standard EA runs while applying standard genetic operators. In the search phase, individuals are crossovered and then submitted to the standard Bit-Flip mutation operator according to a given gene mutation ratio taken from the set of neighborhoods. The process continues until the best individual is a local optimum according to the current neighborhood. This criterion is based on a previous work for Bercachi et al. (7) and used to probably change the current neighborhood. Experimental tests in (7) have shown that a test for a local optimum is the most appropriate criterion for changing the state of solutions. After the local search phase, if the best solution has been enhanced, the algorithm restarts with the first neighborhood. If not, it continues with the following neighborhood. When no far improvement is made, the main effort is to modify the data structure in some way by using different levels of neighborhood structures. Such an approach ensures that new features are introduced into the population. This fact will guarantee a more exhaustive exploration of the search space. Then, an elitist selection operator is applied to keep the so-far best fitness found in the population. Using this mechanism can maintain to the extent possible the sequence of good structures in the population. In VN-EA as in the standard VNS, the order of applying the neighborhoods is predetermined a priori at the beginning of the run. So the same problem of finding the best order of the neighborhoods is also present. For this purpose, we have developed a new approach in which several orders of visiting the neighborhoods are processed simultaneously. Thus, we use the SEA to choose the most adapted order in each time period. Therefore, we have implemented a states-version of VN-EA called States based VN-EA (VN-SEA). It uses diverse VN-EA in a parallel manner depending on the methodology of work of the SEA. VN-SEA benefits from the coexistence of various states. Each state corresponds to a VN-EA with a specific order of the neighborhoods. The VN-EA differ only by the fact that the application of the neighborhoods is performed according to a distinct order each from other. Each VN-EA proceeds in parallel until a predefined number of inner iterations $\text{runPeriod}$, is reached. At this stage, individuals of all states are merged in one
population and undergo a state mutation according to a given state mutation probability \( pMutState \). The application of the changeState operator makes it possible to individuals in the whole population to change their relative states without varying their fitness values. In particular, if during the execution of a given VN-EA no further improvement could be obtained, a state mutation could serve to alternate individuals processing. Therefore, it leads to restart the search process and allows escaping local optima and discovering new regions of the search space. The split-and-merge cycle of VN-SEA keeps on running until a maximum number of total generations or function evaluations is achieved.

3 Description of Experiments

In order to evaluate the new approach and examine the benefits of VN-SEA on a complex problem, we use the MKP. As in many previous publications for the MKP, we use Chu and Beasley’s benchmark library for our tests, which is available at Beasley’s OR-Library \(^1\). We study here the largest instances with \( n = 500 \) items, \( m \in \{5, 10, 30\} \) constraints, and tightness-ratios \( \alpha \in \{0.25, 0.5, 0.75\} \). Each instance has been generated randomly such that \( \forall i = 1, \ldots, m, c_i = \alpha \cdot \sum_{j=1}^{n} w_{ij} \). In our experiments, we have considered a set of nine parameter combinations. Ten different instances exist for each of them yielding 90 instances in total. The values used for all algorithms parameters were chosen as a result of prior experimentations. The best parameter settings between those tested and their relative interpretations are given as follows: VN-SEA is used with three states. A VN-EA is associated to each state. The three VN-EA are similar in everything except they differ in the way the neighborhoods are applied during the search. In the first, second respectively third state, the application of the neighborhood structures is performed according to ascending, descending respectively random order. VN-SEA uses a population size \( popSize \) equal to 10. This choice signifies that MKP requires more exploration than exploitation due to its deceptive attractor. Solutions are encoded with binary strings of length \( n \) equal to 500. The set of neighborhoods is composed by different gene mutation ratios \( k = 10 \). Every time a \( k \)-exchange neighborhoods is occurred, this means that a new bit-flip mutation rate is applied to each individual in the population during the reproduction phase. Gene mutation ratios were varied within a fixed-length interval \([0.01 : 0.1]\) with a step equal to 0.01. Each state will run for a given number of inner iterations \( runPeriod = 1000 \). \( runPeriod \) requires to be affected an enough large value to enable each parallel VN-EA to process sufficiently with its own population before the merge phase takes place. State mutation rate \( pMutState \) has a value of 0.5. A default large value for this parameter was used in such a way that guarantees a kind of equilibrium between the population size in each state. In general, the local exploration in each state is mainly controlled by the application of the changeState operator. VN-SEA stops when a given number of generations \( maxGen = 10^5 \) is attained, producing as well a total number of \( popSize \cdot maxGen = 10 \cdot 10^5 = 10^6 \) function evaluations.

\(^1\)http://people.brunel.ac.uk/~mastjjb/jeb/info.html
### 3.1 Experimental Results

For the purpose of evaluating statistically the results of our proposal, 30 independent runs have been performed using VN-EA\(_a\), VN-EA\(_d\), VN-EA\(_r\) and VN-SEA. VN-EA\(_a\), VN-EA\(_d\) and VN-EA\(_r\) represent a simple VN-EA with the application of the neighborhoods in ascending, descending and random order respectively. Those experiments were performed on the whole set of instances for each of the nine parameter combinations with \(n = 500\) items, \(m \in \{5, 10, 30\}\) constraints, and tightness-ratios \(\alpha \in \{0.25, 0.5, 0.75\}\). The average performance is exhibited in terms of the percentage gap (%-gap). For a solution with a fitness value \(z\), we measure its quality by the \(\%\)-gap = \((z^{LP} - z)/z^{LP}\) . 100%, where \(z^{LP}\) is the solution value of the MKP’s LP-relaxation. Optimal values of the MKP’s LP-relaxation for all cases are available at Beasley’s OR-Library cited in Section 1. Obtained results for all of the aforementioned problem instances are reported in Table 1 with the highest performance scores in bold. In Table 1, data are compared to the most recent related algorithms V&V and W&H which are used to solve the same set of MKP’s problem instances described in this paper. Student \(p\)-value records represent how much the difference between the data of our proposal and those of the best recent algorithm is significant.

### 4 Discussion and Conclusion

We have introduced a new hybrid variant of VNS and SEA. This approach is composed by different states represented by simultaneous VN-EA which are similar in everything except in the ordering of the neighborhoods. The results displayed in Table 1 show that VN-SEA has rendered high positive results in comparison to other algorithms. They also show that VN-SEA outperforms a simple VN-EA with a fixed ordering of the neighborhoods. This can be explained by the fact

---

Table 1: Performance Comparison: VN-SEA vs. Related Algorithms.

<table>
<thead>
<tr>
<th>(m)</th>
<th>(\alpha)</th>
<th>V&amp;V</th>
<th>W&amp;H</th>
<th>VN-EA(_a)</th>
<th>VN-EA(_d)</th>
<th>VN-EA(_r)</th>
<th>VN-SEA</th>
<th>(%)-gap</th>
<th>(%)-gap</th>
<th>(%)-gap</th>
<th>(%)-gap</th>
<th>(p)-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.25</td>
<td>0.07276</td>
<td>0.07185</td>
<td>0.08021</td>
<td>0.09472</td>
<td>0.08276</td>
<td>0.06573</td>
<td>&lt;0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.03784</td>
<td>0.03784</td>
<td>0.04237</td>
<td>0.04866</td>
<td>0.03881</td>
<td>0.02998</td>
<td>&lt;0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.02365</td>
<td>0.02365</td>
<td>0.02944</td>
<td>0.03756</td>
<td>0.03121</td>
<td>0.02164</td>
<td>&lt;0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.25</td>
<td>0.16601</td>
<td>0.17434</td>
<td>0.21732</td>
<td>0.21961</td>
<td>0.21528</td>
<td>0.20649</td>
<td>0.001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.07810</td>
<td>0.07993</td>
<td>0.11389</td>
<td>0.10699</td>
<td>0.12749</td>
<td>0.07786</td>
<td>&lt;0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.05622</td>
<td>0.06562</td>
<td>0.06105</td>
<td>0.05954</td>
<td>0.06154</td>
<td>0.05093</td>
<td>&lt;0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.25</td>
<td>0.48258</td>
<td>0.49695</td>
<td>0.57656</td>
<td>0.59237</td>
<td>0.58451</td>
<td>0.49835</td>
<td>0.028</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>0.20998</td>
<td>0.21740</td>
<td>0.28591</td>
<td>0.26187</td>
<td>0.27309</td>
<td>0.20176</td>
<td>0.089</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.75</td>
<td>0.13511</td>
<td>0.13957</td>
<td>0.19034</td>
<td>0.19924</td>
<td>0.19253</td>
<td>0.14391</td>
<td>0.076</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
that VN-SEA has allowed to the appropriate state to run with a large number of individuals for the most of generations. Hence, the algorithm has advantageously favored the most befitting state. From the other side, it can interpreted that various VN-EA were executing in parallel and shared data between them in favor of the state mutation operator. Thus, VN-SEA uses several potential choices of neighborhood ordering. It can be of great importance for the algorithm process to make the whole search space reachable. Additionally, compounding assorted algorithms can be effective to identify good areas of the search space as fast as possible. Experimental results are confirmed by using the error probabilities measurement displayed in Table 1. For five out of nine MKP set of instances, computed p-value records are very small. So, the difference between the averages of VN-SEA and V&V/W&H is obviously significant. The experiments carried out in this paper corroborate that, for different classes of optimization problems, a hybrid method between VNS and SEA can improve the results achieved by using VNS or SEA separately. Note that combining metaheuristics in a parallel and dynamic model is a highly promising research area for its own and for the adaptive manner that offers to the algorithm operation. We hope that our approach can also be seen as an approving contribution to this direction.

References