Graphical Abstract

Actor-based Large Neighborhood Search

Christian Brunbjerg Jespersen

Highlights

Actor-based Large Neighborhood Search

Christian Brunbjerg Jespersen

- How to allow direct and real-time integration into an optimization process?
- How to perform optimmization in a real-time changing parameter space?

Actor-based Large Neighborhood Search

Christian Brunbjerg Jespersen

[a], , , , ,

Abstract

Abstract text.

Keywords: Large Neighborhood Search, Actor Framework, Real-time Optimization, human-centered computing, Interactive Systems and Tools, Decision Support Systems, Interactive Optimization

1. Introduction

Many problems in operation research are hard to solve due to the need of tight integration with tacit knowledge found in the minds of key decision makers that have to make decisions based on solutions provided by operation research methods such as metaheuristics and commercial solvers. To solve operation research problems that are highly operational in nature there three additional requirements for metaheuristics to have: 1. responsive, 2. able to be assimilated into the decision makers workflow, and 3. allows for integration into dynamic data source such as traditional databases (Meignan et al., 2015). Operational aspects of operation research, as opposed to strategic and tactical aspects, are often characterized by by extensive amounts negotiation and feedback on proposed solutions to general operations research problems (!!!). This can lead to a situation where operation research methods are not used to solve the problem of their underlying model and instead provides intial suggestions Meignan et al. (2015). In practice operations are usually characterized by extensive negotiation and feedback on proposed ”solutions” between relevant stakeholders.

This paper proposes a solution method that will allow for real-time optimization based on user-interaction (changes to the parameter space) and connection to a dynamic data source (in practice a database connection). The proposed solution method will be based on the multi-compartment multi-knapsack problem on a large dataset. Further the large neighborhood search metaheuristic has been chosen due to its properties of being able to work with and fix infeasible solutions.

The paper is divided into four different sections. Section 2 explains the model and method in detail that form the fundation of the paper. Section 3 shows that results coming from the implemented system where the system will be affected by simulated user-interaction. Section 4 will discuss the implications of the research and possible future research directions.

1.1. The Multi-compartment Multi-knapsack Problem with capacity penalties

A variant of the knapsack problem is applied to the actor-based large neighborhood search. It was chosen due to its simplicity while also being sufficiently computationally hard to be able to illustrate the power behind the solution approach. The model is comprised of five different sets. Here: K is the number of knapsacks; I is the number of items; C is the number of compartments; Q is a set that defines which items should be excluded from a specific knapsack; P is an inclusion set that defines the items which should be included in each knapsack. The model introduces four different parameters. Here: v_{ik} is the value associated with item i in knapsack k; d is the penalty paid for exceeding the compartment capacity; w_{ic} is the capacity requirement for item i in compartment c; cap_{kc} is the total amount of capacity available in knapsack k for compartment c. The decision-variables are: x_{ik}, which is a binary decision variable equal to one if item i is in knapsack k and zero otherwise; p_{kc} is non-negative decision variable equal to the amount of excess capacity above the c_{kc} in knapsack k for compartment c.

$$\text{Min} \quad \sum_{i=1}^{I}\sum_{k=1}^{K} v_{ik} \cdot x_{ik} + \sum_{k=1}^{K}\sum_{c=1}^{C} d \cdot p_{kc} \tag{1}$$

subject to:

$$\sum_{i=1}^{I} w_{ic} \cdot x_{ik} \leq \; cap_{kc} + p_{kc} \qquad \forall k \in K, \forall c \in C \tag{2}$$

$$\sum_{i=1}^{I} x_{ik} = 1 \qquad \forall k \in K \tag{3}$$

$$x_{ik} = 0 \qquad \forall (i,k) \in Q \tag{4}$$

$$x_{ik} = 1 \qquad \forall (i,k) \in P \tag{5}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in I, \forall k \in K \tag{6}$$

$$p_{kc} \in \mathbb{R}^{+} \qquad \forall k \in K, \forall c \in C \tag{7}$$

Here the objective function 1 aims to minimize the total weight of all item set assignments under a penalty d given for exceeding the capacity given by constraint 2. Constraint 2 makes sure that all the weights w_{ic} for each item in an item set, given that it has been assigned, is lower than the capacity for each knapsack k and for each compartment c. p_{kc} is the amount of exceed capacity that is needed for the current assignment of item sets to be feasible. Constraint 3 makes sure that each item set is assigned to atleast a single knapsack. Constraint 4 excludes item sets from certain knapsacks and constraint 5 forces a specific item set to be in a specific knapsack. Constraint 6 and 7 specify the variable domain for x_{ik} and p_{kc} respectively. In this paper the effects of changing Q, P, cap, and v in real-time will be examined to determine their effects on the solution and objective value.

2. Solution Method

2.1. Actor-based Large Neighborhood Search

A problem that is affected by user-interaction and requires real-time feedback you need an optimization approach that is able to repair infeasible solutions and while also converging quickly. For this the large neighborhood search metaheuristic has been shown satisfy these requirements in the literature Gendreau and Potvin (2019).

The LNS metaheuristic is in its most basic form is defined for static problems, meaning that the parameters that make up the problem instance is not subject to change after the algorithm has been started. To make the LNS able adapt to incoming information in real-time a message system have been implemented on top of the existing framework. This extension is shown in algorithm 1. In the pseudocode the x

Algorithm 1 Actor-based Large Neighborhood Search

Input queue = message queue
Input P = problem instance
Input x = initial solution
while true **do**
    while queue.has_message() **do**
        $m.update(P)$
        $m.destruct(x)$
    end while
    $x^t = repair(x)$
    if $c(x^t) < c(x)$ **then**
        $x = x^t$
        queue.send(x)
    end if
    queue.push(m)
end while
return x^b

The basic LNS setup have here been extended with a 'message queue'. This message queue will be read from on every iteration of the LNSs main iteration loop. And here we should notice that the incoming message are able to change both the solutoin but also the problem instance itself. Here we see one of the defining features of the LNS metaheuristic in play, that due to its inherrent property of being able to optimize a solution that have become infeasible which is something that is very likely to happen when you change the parameter of the problem instance itself.

Another less obvious property the message queue allows is for the algorithm to run indefinitely and instead of restarting the algorithm you instead pass messages to it to allow it be adjust both the solution space and the parameter space. This property avoid the issue of time consuming initial convergence as the algorithm will be found in an optimimal state when the solution is perturbed.

3. Results

This results section will 1. introduce the data instance 2. show the effect of forcing item set in the specific knapsacks 3. show the effect of changing the knapsack capacities, and 4. show the effect of dynamically changing the item set weights.

3.1. Data Instance

	Number of Item Sets	Number of Compartments	Number of Knapsacks
Instance 1	3487	16	52

Table 1: Table Caption

3.2. Response to Inclusion

The response to the inclusion of a work order is given by P parameter of the model which is constrained in 5 of model given in 1.1.

The inclusion is made of forcing certain allocations of item sets to be in specific knapsack. Below a table is provided to show what changes will occur and at what and at what point in time.

	At Time: 01:00	At Time: 02:00	At Time: 03:00	At Time: 04:00	At Time: 05:00
$\Delta\lvert P\rvert$	10	20	30	40	50

With the inputs defined we will explain the main results which are shown in the figure below.

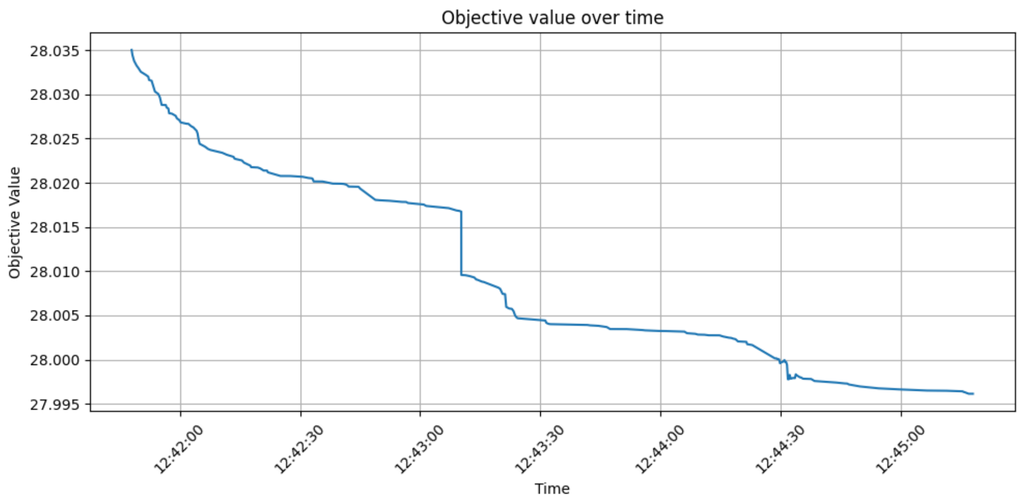

Figure 1: Figure Caption

3.3. Response to Exclusion

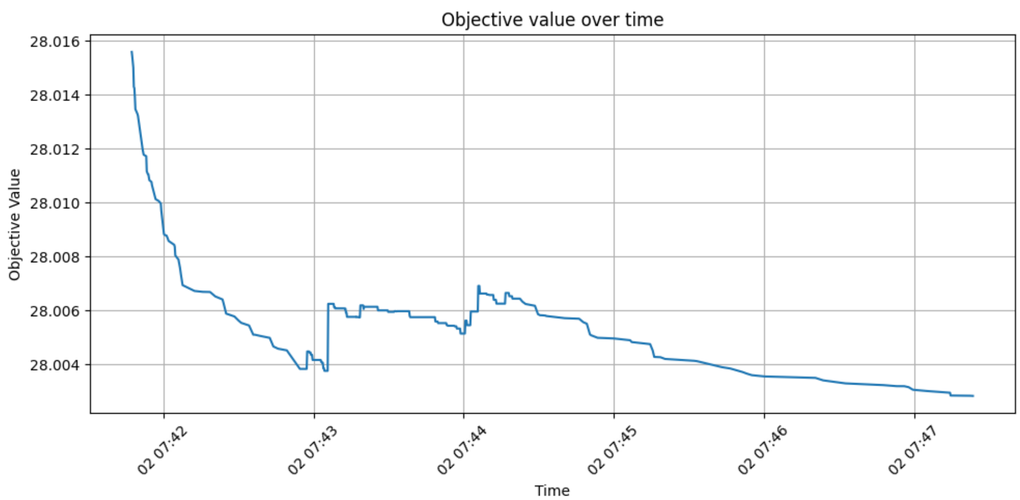

Figure 2: Figure Caption

3.4. Response to Changes in Knapsack Capacities

The effects of changes to capacities will be illustrated in the same way as it was with the response to inclusion and below we see the table that shows which inputs that the ALNS will be affected by.

	At Time: 01:00	At Time: 02:00	At Time: 03:00	At Time: 04:00	At Time: 05:00
$\Delta\|k\|$	16	16	16	16	16
$\Delta\|c\|$	16	16	16	16	16
$\Delta\|cap\|$	100	200	400	800	1600

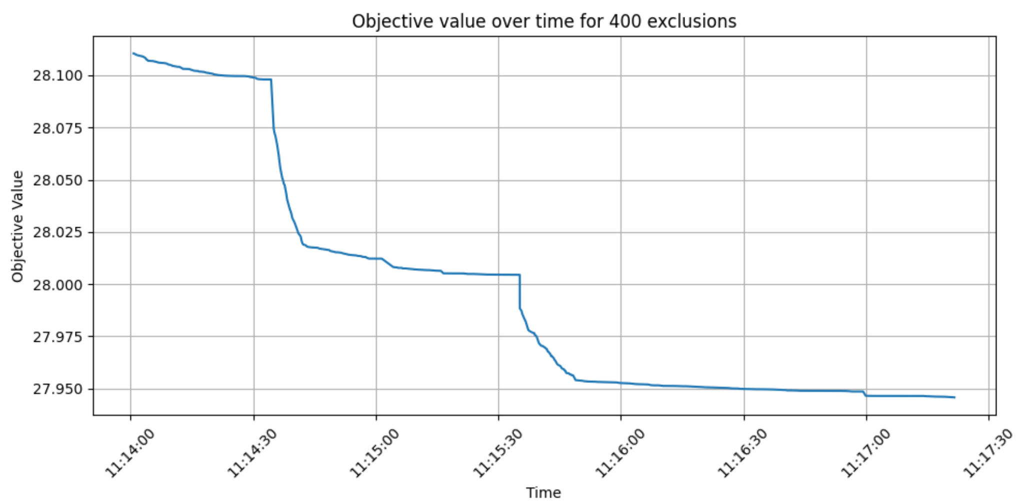

Figure 3: Figure Caption

Correspondingly we also have the figure below in which the resources are decreasing.

3.5. Response to Changes in Item Weights

The last parameter that will be changed in the item set wights. This section will be more elaborate as we have to show how that the item sets are rearranged due to the changes in their weights across the different periods.

	At Time: 01:00	At Time: 02:00	At Time: 03:00	At Time: 04:00	At Time: 05:00
$\Delta\lvert i\rvert$	20	40	80	160	320
$\Delta\lvert k\rvert$	26	26	26	26	26
$\Delta\lvert v\rvert$	$1 \cdot 10^5$	$2 \cdot 10^5$	$4 \cdot 10^5$	$8 \cdot 10^5$	$1.6 \cdot 10^6$

4. Discussion

The key novelty in adopting actor-based large neighborhood search is that it enables tight integration with either users, dynamic data sources, and other models. In figure 4 we see a normal use case of optimization approaches where the user extracts data out from a source which is then supplied to a dedicated optimization algorithm. If an algorithm in this setup is not built to support real-time connectivity only one user will be allowed to use the algorithm at a time for a given set of data. In certain settings, and especially in highly operational ones, this setup can become unworkable, due to many things changing continuously.

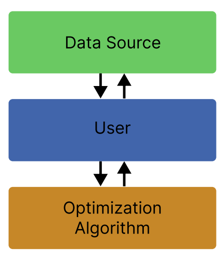

Figure 4: Traditional use case of developed optimization algorithms. Notice that the optimization algorithm is separated from the data source, requiring user interaction to optimize the process

Figure 5 illustrates the setup that is enabled by using an actor-based optimization approach. Due to the implemented message system it becomes possi

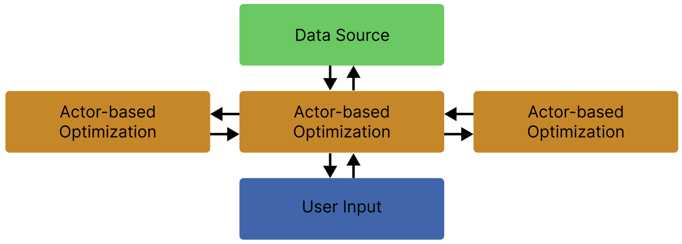

Figure 5: Figure Caption

The three most conseqential properties of this approach is: 1. continual optimization saves the initial time it takes to reach converence significantly, 2. due to the message system the optimization approach will always be responsive to changes in both the parameter space and the solution space, and 3. due to the encapsulation and message passing properties of the suggested optimization approach it becomes possible to apply the metaheuristic in multi-model and hierarchical model setups, providing an approach for modelling and optimizing large scale systems. Finally a suggestion for future resarch directions will be given.

4.1. Continuous Optimization

By continually optimizing it becomes possible to only optimize the pertubations to the schedule. This means that we avoid having to optimize the problem to reach initial convergence.

In many cases this can save a significant amount of computations, especially if specilized constructors and destructors are implemented to handle the specific pertubations.

4.2. Message Passing versus Restarts

In software development when implementing parallel or concurrent programs there is a move towards message passing instead of shared memory.

- Dynamic
- Resource efficient
- Integration (microservice)

4.3. System Level Optimization

References

Gendreau, M., Potvin, J.Y. (Eds.), 2019. Handbook of Metaheuristics. volume 272 of *International Series in Operations Research & Management Science*. Springer International Publishing, Cham. URL: `http://link.springer.com/10.1007/978-3-319-91086-4`, doi:10.1007/978-3-319-91086-4.

Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N., 2015. A Review and Taxonomy of Interactive Optimization Methods in Operations Research. ACM Transactions on Interactive Intelligent Systems 5, 1–43. URL: `https://dl.acm.org/doi/10.1145/2808234`, doi:10.1145/2808234.