Graphical Abstract

Actor-based Large Neighborhood Search

Christian Brunbjerg Jespersen, Thomas Jacob Riis Stidsen, Kristoffer Sigsgaard Wernblad

Highlights

Actor-based Large Neighborhood Search

Christian Brunbjerg Jespersen, Thomas Jacob Riis Stidsen, Kristoffer Sigsgaard Wernblad

- 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, Thomas Jacob Riis Stidsen, Kristoffer Sigsgaard Wernblad

[a] *Technical University of Denmark, Anker Egelundsvej 1, Kongens Lyngby, 2800, Hovedstaden, Denmark*
[b] *Technical University of Denmark, Anker Egelundsvej 1, Kongens Lyngby, 2800, Hovedstaden, Denmark*
[c] *Technical University of Denmark, Anker Egelundsvej 1, Kongens Lyngby, 2800, Hovedstaden, Denmark*

Abstract

Serveral problems facing the operations research field have proven difficult to solve due to their inherrent uncertainty and highly dynamic nature. Stochastic optimization, fuzzy logic, and robust optimization are some of the methods that have been proposed to solve these issues. These methods make an implicit assumption on static data and a static problem setting. This paper will argue that a new class of optimization methods will have to be developed by reflect the need optimizing in a settings where: the data source is changing in real-time; where external inputs affects the optimization process; where multiple actors are making interdependent decision whose objectives may differ significantly. This paper proposes an actor-based implementation of the classic large neighborhood search metaheuristic as a specific contribution to optimization approachs that more naturally model the dynamic nature of many operational problems.

Keywords: Large Neighborhood Search, Actor Framework, Real-time Optimization, Human-centered Computing, Interactive Systems and Tools, Decision Support Systems, Interactive Optimization.

1. Introduction

Dynamic and operational problems have proven hard to solve in operation research due to the need of tight integration with tacit knowledge of decision makers and the way that industry usually assigns responsibility for decision-making to an individual representing only a small part of the complete process. These many smaller parts are often difficult to map to a single mathematical model describing the whole system as elaborated by ((Barthélemy et al., 2002)). Solving operation research problems that are highly operational in nature have additional requirements over conventional static solution approaches: they have to be responsive to changing parameters; able to be assimilated into the decision-makers workflow; allow for integration with dynamic data sources such as databases and RESTapi (Meignan et al., 2015). Operational aspects of operation research, as opposed to higher level strategic and tactical aspects, are characterized by extensive amounts negotiation and feedback on proposed solutions. The lack of integration and responsiveness can lead to solutions that are not directly implemented in practice but instead provides intial suggestions Meignan et al. (2015), which are then iterated on later manually. In (Barthélemy et al., 2002) the authors argue that many problems that operation research aim to solve are often composed of a group of individuals whose decisions are consolidated into an "epistemic subject" for which a mathematical model can be formulated and solved, with many scheduling problems being good examples. Furthermore some multi-objective optimization problems are a product of there being multiple actors in the decision making process each with different views on an optimal solution from their vantage point rather then there being actual multi-objective for an individual actor.

This paper proposes a solution method that will allow for real-time optimization based on actor/user interaction and connection to a dynamic data source, effectively meaning changes to the parameter space. The proposed solution method will be tested on the multi-compartment multi-knapsack problem (MCMKP) on a large dataset. The solution method will by based on the large neighborhood search (LNS) metaheuristic. This meta heuristic was chosen due to its properties of naturally being able to work with and fix infeasible solutions and its state of the art performance on various scheduling problems.

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

The actor-based large neighborhood search is implemented on the MCMKP. The MCMKP was chosen due to its simplicity while also being sufficiently computationally hard to illustrate the principle behind the Actor-based LNS. The model is comprised of five different sets. 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 allocation of specific item sets which should be included in a specific knapsack. The model has four parameters. v_{ik} is the value of item i in knapsack k; d is the penalty for exceeding 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 model has 2 decision variables. x_{ik}, 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. The parameters v, cap, Q, and P are functions of time, t, in this case as they will be subject to change during the solution process.

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

subject to:

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

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

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

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

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

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

The objective function 1 minimizes the total weight of all item set assignments together with the penalty d for exceeding the capacity given in constraint 2. Constraint 2 ensures 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 exceeded 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. 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 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 changing parameters 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

2.1.1. Messages And Destructors

LNS in its most basic form has one constructor and one destructor which repeatedly destroy and rebuild the solution. For the AbLNS we will generalize on this concept by including messages as destructors of the classic LNS implementation. This generalization can be seen as being somewhat similar to how the adaptive LNS (ALNS) is formulated.

Extending on the classic setup we define the following destructors:

- m_1: Inclusion destruct message
- m_2: Exclusion destruct message
- m_3: Capacities destruct message
- m_4: Weights destruct message
- m_5: Random destruct message

Each of these messages affect different parts of the MCMK problem. Notice here that the first four messages destruct the solution by changing the parameter space and the last message is a random destructor.

Generalizing the destructors from being static structures into messages allows the solution to change in real-time to a changing paramenter space meaning that the algorithm does not need to restart to handle changes in data.

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**
        $P.update(m)$
        $x.destruct(m)$
    end while
    $x^t = x.repair()$
    if $c(x^t) < c(x)$ **then**
        $x = x^t$
        queue.send(x)
    end if
    queue.push(m_5)
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. Here we 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
$\Delta\|P\|$	10	20	30	40	50

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

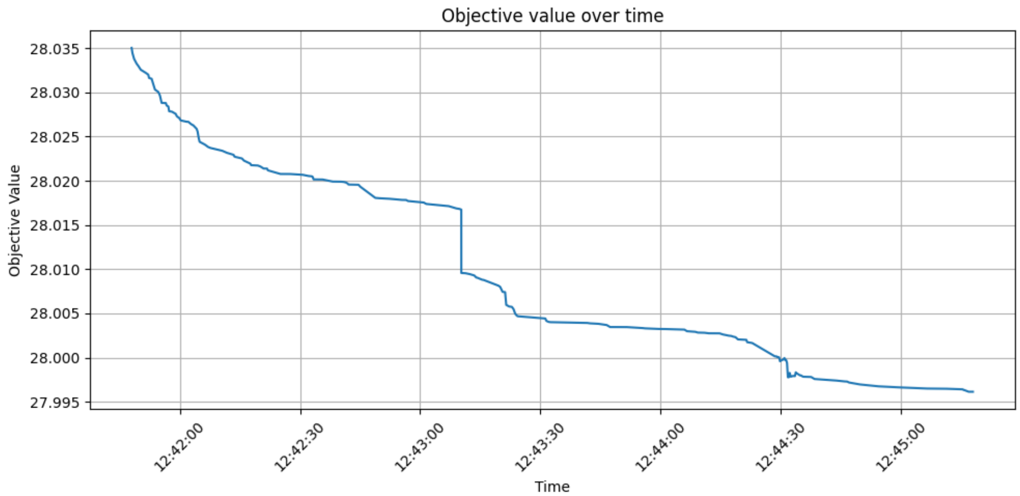

Figure 1: Figure Caption

3.3. Response to Exclusion

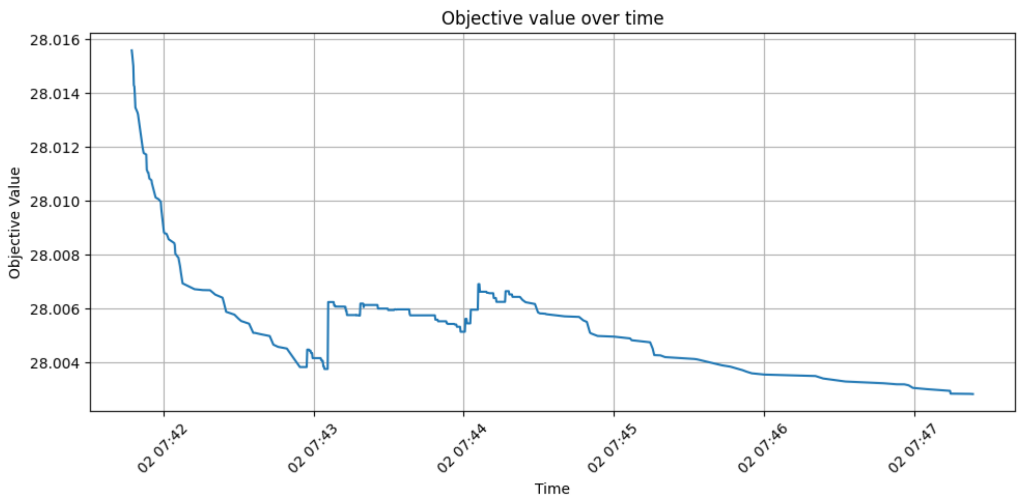

Figure 2: Figure Caption

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

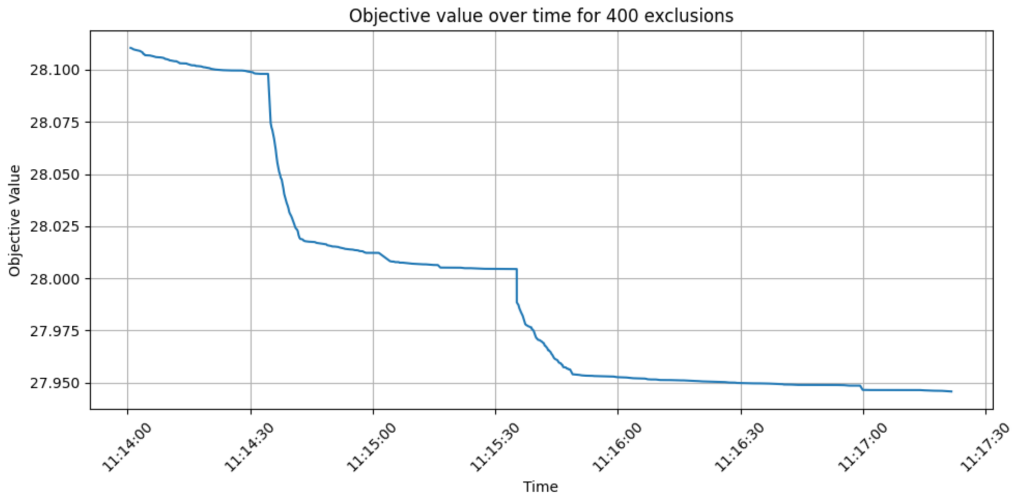

Figure 3: Figure Caption

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

3.5. Response to Changes in Item Weights

The final parameter that will be changed is the item set weights. 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
$\Delta\|i\|$	20	40	80	160	320
$\Delta\|k\|$	26	26	26	26	26
$\Delta\|v\|$	$1 \cdot 10^5$	$2 \cdot 10^5$	$4 \cdot 10^5$	$8 \cdot 10^5$	$1.6 \cdot 10^6$

4. Discussion

4.1. Integration

Figure 4illustrates a classic approach to operations research where a model allows the decision maker the obtain a solution to from which to make better informed decision. This approach is suitable to static problems and problems there the parameters that make up the problem change slowly. This approach is often unsatisfactory in environments where the environment is ever changing and the model parameters continually change in that light of new information. To solve this problem a new field of dynamic and interactive metaheuristics show promise.

Actor-based large neighborhood search enables tight integration between both the users and a dynamic data sources but crucially it naturally allows models to communicate with each other mimicking most practical decisions which are made up of multiple actors. For example, maintenance scheduling problems.

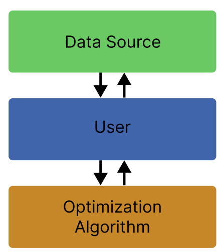

Figure 4: Traditional use case of developed optimization algorithms. Notice that the optimization algorithm is separated from the data source, requiring user interaction to optimize the process

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

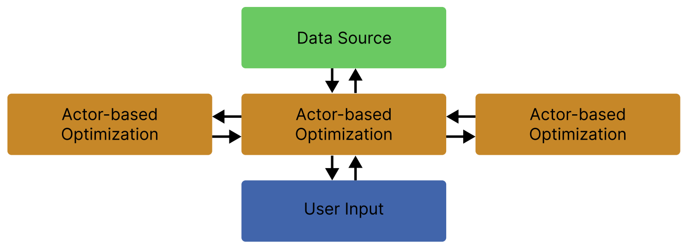

Figure 5: Figure Caption

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.

4.2. 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, refer to alza et al. (2023), especially if specilized constructors and destructors are implemented to handle the specific pertubations.

4.3. Message Passing versus Restarts

There is evidence that the effect of dynamic optimization is not always efficient, alza et al. (2023) mentions the idea that dynamic optimization approaches can be "elusive" in that they do not always provide a speedup over restarting the solution method implementation. The paper clearly shows that dynamic approaches are the most effective when changes are small and frequent, which does align with the idea behind the actor-based LNS in that changes should be optimized around when ever they occur. T

4.4. System Level Optimization

This approach could also enable larger scale optimzations providing a modular approach to operations research. This is beneficial

References

alza, j., bartlett, m., ceberio, j., mccall, j., 2023. on the elusivity of dynamic optimisation problems. swarm and evolutionary computation 78, 101289. URL: https://linkinghub.elsevier.com/retrieve/pii/s2210650223000627, doi:10.1016/j.swevo.2023.101289.

Barthélemy, J., Bisdorff, R., Coppin, G., 2002. Human centered processes and decision support systems. European Journal of Operational Research 136, 233–252. URL: https://linkinghub.elsevier.com/retrieve/pii/S0377221701001126, doi:10.1016/S0377-2217(01)00112-6.

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.