Computers and Chemical Engineering 176 (2023) 108282

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/cace

Data-driven derivative-free trust-region model-based method for resource allocation problems

Joakim R. Andersen [a,*], Lars Imsland [a], Alexey Pavlov [b]

[a] Department of Engineering Cybernetics, Norwegian University of Science and Technology, O. S. Bragstads Plass 2D, Trondheim, 7034, Norway
[b] Department of Geoscience and Petroleum, Norwegian University of Science and Technology, S. P. Andersens veg 15a, Trondheim, 7031, Norway

ARTICLE INFO

Keywords:
Resource allocation problem
Derivative-free trust-region method
Data-driven optimization
Problem structure exploitation

ABSTRACT

Allocating a limited available resource between a set of units is a problem that arises in several application areas. We propose an online derivative-free trust-region model-based method to tackle a fairly general version of the resource allocation problem where units may be turned on or off. The units are considered as black boxes which may only be evaluated given that all the other units are evaluated simultaneously, and no gradient information is available. This method was inspired by an industrial problem and emphasis is put on both providing feasible points during the optimization and on not incurring additional increase in cost while searching for the optimum. The latter cannot be guaranteed, but the algorithm allows for automatic or manual ranking of the different units to attempt to reduce negative impact on the cost. The algorithm was applied to a case study from the petroleum industry where fast convergence was observed.

1. Introduction

The resource allocation problem is concerned with optimally allocating a limited available resource between a set of units to minimize (or maximize) a cost (or reward) function. This type of problem arises in different application areas such as queuing control, computer resource allocation, apportionment, load distribution, portfolio selection and production planning (Katoh et al., 2013). Several different variations of the resource allocation problem exists. The formulation studied in this work is mathematically formulated as:

$$\min_{\mathbf{u},\mathbf{z}} \quad f(\mathbf{u}) = \sum_{i=1}^{n_u} f_i(u_i) \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{1b}$$

$$z_i \underline{u}_i \leq u_i \leq z_i \overline{u}_i \tag{1c}$$

$$u_i \in \mathbb{R} \tag{1d}$$

$$z_i \in \{0,1\} \tag{1e}$$

where $u_{\max}$ is the available resource, u_i is the amount of resource allocated to unit i, and $f_i(u_i)$ is the cost of allocating u_i to unit i. Bold notation is used for vectors: $\mathbf{u} = [u_1, u_2, \dots, u_{n_u}]^\top$ and $\mathbf{z} = [z_1, z_2, \dots, z_{n_u}]^\top$. If all $z_i = 1$, then this would be a standard problem formulation, as found in Katoh et al. (2013). The binary variables allow for turning on and off units if necessary to achieve a better cost. Throughout this work, we assume that the f_i's are convex, and thus also f. Should the f_i's be non-convex, we rely on the generally accepted, though not proved, statement that derivative-free methods typically have the ability to find good local minima due to their relative crudeness, see Conn et al. (2009). Moreover, we assume the following characterizes f_i:

(i) the f_i's are unknown functions, and can only be evaluated and no gradient information is available,
(ii) f_j can only be evaluated if all the other f_i's are evaluated simultaneously,
(iii) all the constraints must be satisfied to perform an evaluation, and
(iv) evaluating f is costly.

The reason for the latter, in addition to the actual cost of measurement, could be that evaluating f involves a change of operation point with potential loss of revenue, and also, if there are underlying dynamics, that reaching a new steady-state after changing the allocation may be a slow process. The term *true function* is used to refer to the unknown cost function $f(\mathbf{u})$. In this work, the resource allocation problem is formulated as a minimization problem where the objective function is a metric of the cost of allocating the resource. If the true function is the revenue of allocating the resources instead of the cost, the objective function should simply be multiplied by −1 to achieve a maximization problem.

* Corresponding author.
E-mail addresses: joakim.r.andersen@ntnu.no (J.R. Andersen), lars.imsland@ntnu.no (L. Imsland), alexey.pavlov@ntnu.no (A. Pavlov).

https://doi.org/10.1016/j.compchemeng.2023.108282
Received 19 August 2022; Received in revised form 29 December 2022; Accepted 29 April 2023
Available online 4 May 2023

We propose a customized derivative-free trust-region model-based method to solve the problem (1) taking the characteristics (i)–(iii) into account. In addition, consideration (iv) is dealt with by introducing logic to try to reduce the negative impact on the cost.

In derivative-free trust-region model-based optimization it is, typically, assumed that the to-be-optimized function, or *true function*, is unknown. This function can be seen as a black box as it may only be evaluated, and no gradient information is available. A standard approach in these methods is to build a first or second order polynomial model of the true function using the gathered evaluations. This *surrogate model* is trusted within some *trust-region* typically defined by a *center point* and a *trust-region radius*. The surrogate model of the cost is minimized within the trust-region, and the true function is evaluated at this point. Assuming that the prediction of the model is good, the point is accepted as the new center point and the process is repeated. If the model prediction is bad, steps must be taken to improve the surrogate model.

The most distinct feature of the proposed algorithm is that it has as many trust-region radii and models as there are units. Each unit is treated individually when it comes to modeling. Furthermore, the algorithm only provides feasible point with respect to the constraints of the problem (1).

To the best of the authors' knowledge, this is the first work to explicitly tackle the resource allocation problem with a (tailored) derivative-free trust-region model-based method. In addition, no previous literature on applying a derivative-free trust-region method on resource allocation problems, nor on such problems with units being considered black boxes, were found. Of course, there is a significant body of literature on more general black box optimization formulations, including Conn et al. (2009), Powell (1994, 2009), Bajaj et al. (2018), Bajaj and Hasan (2019).

The proposed algorithm is applied to an example coming from the oil and gas industry. The goal is to distribute the available lift gas (the resource) between different wells (the units) to maximize the oil production (the revenue). The example is detailed in Section 4. This problem has been studied in several papers, see *e.g.*, Krishnamoorthy et al. (2016a,b), Peixoto et al. (2015), Rashid (2010), Rashid et al. (2012) and the references therein. Differently from these, we focus on the setup where there is no available model of the relationship between the allocated resources and the resulting costs. *I.e.*, the wells, or units, may be viewed as black boxes. Further, only steady-states are considered and no interactions between the wells (through the reservoir) are considered. Moreover, the decision of closing and opening wells are included. The resulting algorithm is a data-driven optimization method.

The task of distributing available lift gas to optimize oil production have been previously tackled by derivative-free trust-region model-based methods (Giuliani et al., 2013; Giuliani and Camponogara, 2015b,a). These papers consider a similar setup to (1), *i.e.*, the minimization of a function subject to linear constraints, except for Giuliani and Camponogara (2015a) which in addition handles nonlinear constraints. Further, they are not considering the option of measuring the output of each unit independently. Problem structure is exploited in Giuliani and Camponogara (2015b) to reduce the amount of required points to build a quadratic model. Our proposed method differs from the previous approaches (Giuliani et al., 2013; Giuliani and Camponogara, 2015b,a) in several aspects. First, we include the decision of turning on and off units. Second, the method will only provide feasible points to evaluate. The constraints are satisfied both when finding a new potential best point, and when points are found to improve the geometry to construct new models. Third, the assumed availability of the individual output of each unit allows for further exploitation of problem structure. The latter is the reason why the suggested approach has as many trust-region radii and models as there are units.

In the context of the gas lift optimization problem, we propose a decomposition of the gas lift performance curve commonly used in the distribution of the available lift gas. The decomposition allows for efficient exploitation of the available data. To the best of the authors' knowledge, this decomposition has not previously been used to exploit data in such a manner.

This paper is structured as follows. In Section 2, the algorithm is developed and detailed. In Section 3, an extension to the algorithm is presented that exploits unused degrees of freedom to attempt reducing negative impact on the cost. In Section 4, the motivational example is detailed, and the algorithm is applied to a small example. Furthermore, the section contains details on the proposed decomposition. In Section 5, several aspects of the algorithm is investigated by using different parameters of the algorithm and different instantiations of the motivational example. In Section 6, a discussion on the algorithm is provided. Finally, in Section 7 a conclusion is provided.

2. Theory

In this section, the proposed algorithm will be presented. It starts by introducing all the different parts of the algorithm before they are assembled into a complete algorithm at the end. The different buildings blocks of the proposed derivative-free trust-region model-based method are inspired by the framework presented in Conn et al. (2009). The method differs from this framework in several ways, which will be highlighted and discussed as each building block is presented.

2.1. Surrogate modeling

The suggested algorithm was tailored for a setup where the quality of the input–output relationships of the units were prone to measurement errors and measurements are few. In a noise free environment, quadratic surrogate models would typically allow for faster convergence. However, we use linear models to mitigate the impact of the noise. Furthermore, the typical samples would be rather close to each other making the extrapolation capabilities of a higher order polynomial inferior. Each unit i will have its own linear surrogate model F_i of the true input-to-cost relationship f_i

$$F_i(u_i) = a_i u_i + b_i \tag{2}$$

where the parameters a_i and b_i are chosen such that F_i either match the collected evaluated points, or minimizes the squared prediction error at these points if regression is used.

It is important to take care of the spread, or geometry, of the points that are used for finding the parameters a_i and b_i. For example, say there are two points used for creating the linear model. First, if the two points lie on top of each other, any line through the point would be a perfect interpolation, but it is clearly of no use. Second, assume there is a trust-region radius of 10^{10} centered at the origin. Two points located at $u_i = 0$ and $u_i = 1$ will have a very small chance of capturing any valuable information of the true function over the entire trust-region, unless it happens to be linear and noise free. With these two examples in mind, the following geometry requirement rules are proposed.

2.1.1. Geometry requirement — interpolation

Let $u_{i,\text{cp}}$ be the current center point of the model for unit i, and let Δ_i denote its trust-region radius. The set $\mathbf{U}_i = \{u_{i,\text{cp}}, u_{i,1}\}$, of length n_i, is deemed poised (or suitable) for interpolation if:

(i) At least one $j \in [1, \dots, n_i - 1]$ satisfies

$$|u_{i,\text{cp}} - u_{i,j}| \geq \frac{1}{2}\Delta_i \tag{3}$$

(ii) All $j \in [1, \dots, n_i - 1]$ satisfies

$$|u_{i,\text{cp}} - u_{i,j}| \leq r\Delta_i \tag{4}$$

(iii) All $j \in [0, \dots, n_i - 1]$ satisfies

$$\underline{u}_i \leq u_{i,j} \leq \bar{u}_i \tag{5}$$

The $r \geq 1$ in (4) is a scaling factor which allows for more reuse of points and is a trick borrowed from Conn et al. (2009). Condition (i) guarantees spread of the points, (ii) guarantees the points are within the (possibly extended) trust-region, and (iii) guarantees the points used for modeling is feasible with respect to the bounds of the unit.

2.1.2. Geometry requirement — regression

In the case when the cardinality of $\mathcal{U}_i$ is greater than two, $n_i > 2$, the model making process may be done through least squares regression instead of interpolation. Let $\mathcal{U}_{i,\text{all}}$ be the set of (all) previously evaluated points. Further, let $\mathcal{U}_i$ be the subset of $\mathcal{U}_{i,\text{all}}$ which contains all the points that satisfies conditions (ii)–(iii). Next, the $\mathcal{U}_i$ is split into two sets: $\mathcal{U}_i^0 = \{u_{i,j} \in U_i : |u_{i,\text{cp}} - u_{i,j}| < \frac{1}{2}\Delta_i\}$ and $\mathcal{U}_i^1 = \{u_{i,j} \in U_i : |u_{i,\text{cp}} - u_{i,j}| \geq \frac{1}{2}\Delta_i\}$. If $\mathcal{U}_i^1$ is a non-empty set, $\mathcal{U}_i$ is deemed poised (or suitable) for regression.

This classification of poisedness for regression is aligned with the one in Conn et al. (2009): A set of bounded (and moderate) amount of points will be poised for regression if a subset is poised for interpolation. It may have a worse "poisedness", but only by a scaling factor.

2.1.3. Filtering

If there is a large imbalance in the cardinalities, the smallest may end up having close to no influence. A solution could be to set a cap on the cardinality of the sets $\mathcal{U}_i^0$ and $\mathcal{U}_i^1$. That is, instead of adding all points satisfying the aforementioned requirement when making $\mathcal{U}_i^0$ and $\mathcal{U}_i^1$, only select up to the $n_{\max}$ newest.

To avoid that restricting the cardinality results in only identical points, which could be the case if the unit is not altered for several iterations, a filtering method should be applied. If the applied filtering method is based on merging the new point with older points, then care must be taken to avoid unwanted effects of the merging. E.g., if a new point is found to satisfy Conditions (i)–(iii), we must ensure that the merged point also satisfies the criteria.

2.1.4. Geometry requirement for units turned off

The optimization problem that should be solved in (1) includes the binary variables to turn on and off units. To respect consideration (iv), evaluating f is costly, the geometry requirement is not imposed for units which are off at the current operation point.

2.2. Geometry improvement

So far, it has been discussed requirements on the sets of points used for interpolation (and regression). If these requirements are not satisfied, the algorithm must prioritize creating such sets of points before continuing looking for the optimum.

Finding the required set of new u_i's to evaluate is not a trivial task due to the interconnection between the units given by the availability of the shared resource (1b). *E.g.*, if one unit needs more resource, one or more of the other units must be allocated less. However, these units have their own lower bounds on the resource requirement (1c). An idea could be to always decrease the use of resource, as then no resource needs to be "borrowed" from another unit. However, the unused resource must be allocated somewhere and the issue of where to route it without hitting the upper bounds arises.

Some terminology and notation will be introduced next. A unit that does not need a new point to pass the geometry requirement is referred to as a *neutral unit* and the index k will be used for these units. Similarly for a unit that needs a point, we will use the term *lacking unit* and index j. Finally, index i is used when it concerns all the units regardless of the geometry requirement.

The new point to improve geometry will be found by solving an optimization problem. Finding a point u_j that satisfies conditions (i)–(ii) for a single unit may be formulated as three constraints using the two binary variables $x_{j,l}$ and $x_{j,r}$:

$$u_j \geq (u_{j,\text{cp}} - \Delta_j)x_{j,l} + (u_{j,\text{cp}} + 0.5\Delta_j + \Delta_=)x_{j,r} \tag{6a}$$

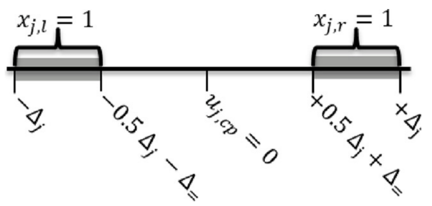

Fig. 1. The feasible areas for the wells that needs new points are shown in gray.

$$u_j \leq (u_{j,\text{cp}} - 0.5\Delta_j - \Delta_=)x_{j,l} + (u_{j,\text{cp}} + \Delta_j)x_{j,r} \tag{6b}$$

$$x_{j,l} + x_{j,r} = z_j \tag{6c}$$

The subscript l and r refer to "left" and "right", respectively. The z_j will be set to one for these units, but is included to keep the notation similar to what will be presented later. Notice that the $\Delta_=$ is included to avoid that the new point will be merged back into the area $u_{j,\text{cp}} \pm 0.5\Delta_j$ if a filtering method based on merging is applied. The feasible areas for u_j with $u_{j,\text{cp}} = 0$ is shown as the shaded gray areas in Fig. 1.

In addition to the new constraints above, the constraint on the lower and upper limit (1c) must be imposed. To help the solver, the on/off variable z_i may be manually set to 1 for the units that need a new geometry improving point.

It should perhaps be noted that finding a single point, a $\mathbf{u}$, that provides all the necessary points may be infeasible. Furthermore, it may be infeasible to find such a point even with only one lacking and one neutral unit. *E.g.*, assume there are two units: $100 \leq u_1 \leq 200$, and $2 \leq u_2 \leq 4$ with a current operating point of $u_1 = 100$ and $u_2 = 2$, and that unit 2 needs another point. Considering that unit 1 is also at its lower limit, we need to shut unit 1, but then there is too much resource to be distributed. Hence, this problem is infeasible by problem definition. This type of infeasibility is hereafter referred to as *problem infeasibility*. Nonetheless, this type of "problem infeasibility" was never encountered in the case study and it is less likely when the amount of units increases. Furthermore, it could be argued that (i) such input problems should be avoided by the user, or (ii) the algorithm could exit with the current operation point as the solution when it occurs.

The resulting optimization, or feasibility, problem for improving the geometry is a Mixed-Integer Linear Program (MILP):

$$\min_{\mathbf{u},\mathbf{z},\mathbf{x}} \quad 0 \tag{7a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{7b}$$

$$u_j \geq (u_{j,\text{cp}} - \Delta_j)x_{j,l} + (u_{j,\text{cp}} + 0.5\Delta_j + \Delta_=)x_{j,r} \tag{7c}$$

$$u_j \leq (u_{j,\text{cp}} - 0.5\Delta_j - \Delta_=)x_{j,l} + (u_{j,\text{cp}} + \Delta_j)x_{j,r} \tag{7d}$$

$$x_{j,l} + x_{j,r} = z_j \tag{7e}$$

$$z_i\underline{u}_i \leq u_i \leq z_i\bar{u}_i \tag{7f}$$

$$u_i \in \mathbb{R} \tag{7g}$$

$$z_j = 1 \tag{7h}$$

$$z_i \in \{0, 1\} \tag{7i}$$

$$x_{j,l}, x_{j,r} \in \{0, 1\} \tag{7j}$$

As mentioned above, the task of finding a single point satisfying (7c) and (7d) for all the lacking units may be infeasible. Provided that the "problem infeasibility" is not the issue, several evaluations of the true function would be required. Let $\mathcal{J}$ be the set of all the j indices, *i.e.*, it contains the indices of all the lacking units. One could, for example, solve (7) two times where the set $\mathcal{J}$ is split into two (equally large) sets

$\mathcal{J} = \mathcal{J}_0 \cup \mathcal{J}_1$. If one (or both) fails, repeat the branching process until feasibility is reached in all branches. For each branching, the required amount of lacking units that receives a new point is decreased and, thus, the optimization problem becomes less constrained.

There may be several $\mathbf{u}$, $\mathbf{z}$ and $\mathbf{x}$ that satisfies the constraints in (7). In Section 3, these unused degrees of freedom will be exploited to try to minimize the increase in cost while performing a geometry improvement step.

2.3. Solving the subproblem

When all units have enough points to create a decent model, it is time to look for a new operation point that should further minimize f. This problem is referred to as the *subproblem* in trust-region literature. Let F_i be the (linear) surrogate model of f_i. The subproblem is given as

$$\mathbf{u}^+, \mathbf{z}^+ = \arg\min_{\mathbf{u},\mathbf{z}} \quad F(\mathbf{u}) = \sum_{i=1}^{n_u} F_i(u_i) \tag{8a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{8b}$$

$$u_i \geq z_i(u_{i,\mathrm{cp}} - \Delta_i) \tag{8c}$$

$$u_i \leq z_i(u_{i,\mathrm{cp}} + \Delta_i) \tag{8d}$$

$$z_i \underline{u}_i \leq u_i \leq z_i \overline{u}_i \tag{8e}$$

$$u_i \in \mathbb{R} \tag{8f}$$

$$z_i \in \{0, 1\} \tag{8g}$$

As opposed to the standard derivative-free trust-region methods, we may employ an off-the-shelf Mixed-Integer Linear Program (MILP) solver to solve the subproblem. A standard method for finding $\mathbf{u}^+$ is to only take a single step that provides sufficient decrease of the objective function, see Conn et al. (2009) for details. If the solution of the subproblem yields a better value of f, then it is used as the next operation point.

2.4. Trust-region radius update

Different from the standard approach in derivative-free trust-region methods, several surrogate models are in use. Each unit will have its own surrogate model and trust-region radius. Let u_i^+ be the point where the model prediction quality ρ_i should be evaluated:

$$\rho_i = \frac{f_i(u_{i,\mathrm{cp}}) - f_i(u_i^+)}{F_i(u_{i,\mathrm{cp}}) - F_i(u_i^+)} \tag{9}$$

The trust-region radius for unit i is updated according to:

$$\Delta_i \leftarrow \begin{cases} \Delta_i & \text{if } \rho_i \geq \eta_1 \\ \max(\gamma \Delta_i, \Delta_{i,\min}) & \text{else} \end{cases} \tag{10}$$

with $0 < \gamma < 1$. In words, if the model prediction quality is deemed good, the radius is kept fixed, otherwise it is reduced.

The functionality for turning on/off units and using a surrogate model for each unit introduces some additional scenarios that must be dealt with. If either,

- the unit is off at the next point (*i.e.*, $z_i^+ = 0$), or
- the unit has never been on, or
- the resource allocation remains the same for the unit, *i.e.*, $u_{i,\mathrm{cp}} = u_i^+$

then do not calculate ρ_i and skip updating the trust-region radius.

Finally, it should perhaps be noted that despite the sets of points used to create the F_i's are all deemed poised, the prediction quality may still be bad. The poisedness only guarantee a certain amount of spread in the points. The aim is that the spread should help getting points that capture valuable information that can be used for modeling. However, there is no (strong) guarantee on the resulting prediction quality.

2.5. Acceptance of new point

A point $\mathbf{u}^+$ found while solving (8) will be accepted as the new operation point as long as:

$$\frac{f(\mathbf{u}_{\mathrm{op}}) - f(\mathbf{u}^+)}{F(\mathbf{u}_{\mathrm{op}}) - F(\mathbf{u}^+)} > 0 \tag{11}$$

Which means that as along as the new point has a lower true function value, it will be accepted as the new operation point.

Notice that the subscript "op" is used to refer to the operation point, and it should not be confused with the center point of the model. Each unit, will have its own model center point. If the unit is on, the center point will be the last solution of (8). If the unit is off, its center point will be the last solution of (8) when it was on, *i.e.*, its z_i was 1. The term operation point is used for the point that is currently the best $\mathbf{u}^+$ point encountered. This means that $\mathbf{u}_{\mathrm{op}}$ will always satisfy the constraints of (1), whereas $\mathbf{u}_{\mathrm{cp}}$ may not. The individual $u_{i,\mathrm{cp}}$ will satisfy the upper and lower input constraints for that unit.

2.6. Criticality step

Another building block for the trust-region method is the *criticality step*. It ensures a relationship between the size of the trust-region radius and a measurement of stationarity (Conn et al., 2009). Its task is to make the models more accurate when this measure is close to zero. This will drive the trust-region radius towards zero.

The criticality step presented in Conn et al. (2009) is intended for the minimization of a function without any constraints. In the unconstrained case, the measure of stationarity is the gradient of the objective function. For the constrained case, we should use the gradient of the Lagrangian instead:

$$\nabla_{\mathbf{u}} \mathcal{L}(\mathbf{u}, \mathbf{z}, \lambda, \boldsymbol{\mu}) = \begin{bmatrix} \nabla_{u_1} \mathcal{L} \\ \nabla_{u_2} \mathcal{L} \\ \vdots \\ \nabla_{u_{n_u}} \mathcal{L} \end{bmatrix} \tag{12}$$

$$\nabla_{u_i} \mathcal{L} = \nabla_{u_i} F(\mathbf{u}) + \lambda - \underline{\mu}_i + \overline{\mu}_i \tag{13}$$

where

$$\mathcal{L}(\mathbf{u}, \mathbf{z}, \lambda, \boldsymbol{\mu}) = F(\mathbf{u}) + \lambda \left(\sum_{i=1}^{n_u} u_i - u_{\max} \right) + \sum_{i=1}^{n_u} \left[\underline{\mu}_i (z_i \underline{u}_i - u_i) + \overline{\mu}_i (u_i - z_i \overline{u}_i) \right]$$

Notice that the constraints limiting the search area to the trust-region (8c) and (8d) are not included. If they were, the gradient of the Lagrangian would simply be zero each time at the optimum of (8).

Using the Lagrangian instead of the objective function introduces several issues which will be discussed next.

The first issue will be illustrated by a small example. Consider that there are two units, and that the current operation point is far away from the lower and upper bounds of each unit. Then we have

$$\nabla_{u_1} \mathcal{L} = \nabla_{u_1} F_1(\mathbf{u}) + \lambda$$
$$\nabla_{u_2} \mathcal{L} = \nabla_{u_2} F_2(\mathbf{u}) + \lambda$$

where the bounds are left out for simplicity. The λ will receive its value based upon the two equations above. Further, assume that the model prediction quality has been found to be good for one unit ($\rho_1 = 1$) and bad for the other ($\rho_2 = 10^{-4}$). As the underlying functions f_i's are assumed to be convex, it is fair to say that a low prediction quality indicates a bad gradient prediction too. If one of the two gradients $\nabla_{u_1} F_1(\mathbf{u})$ and $\nabla_{u_2} F_2(\mathbf{u})$ are incorrect, then the quality, or usefulness, of all the elements of the gradient of the Lagrangian is at stake as they are all intertwined through the λ multiplier. To deal with this issue, the gradient of the Lagrangian is only evaluated whenever $\rho_i \geq \eta_1$ for all the considered units (as explained in Section 2.4).

Another issue when going from the gradient of objective function to the gradient of the Lagrangian is where to evaluate the gradient. In Conn et al. (2009), the gradient of the model is evaluated at the center point of the model, or the operation point in our setting. The Lagrange multipliers in (13) complicate this choice. The multipliers that satisfies the First Order Necessary Conditions (FONC) of optimality (Nocedal and Wright, 2006) are only available at a local optimum. However, the current operation point will rarely (or never) be such a point. Once the operation points for the unit models are moved, so will (most likely) the solution of (8), even though none of the F_i's are updated. The reason for the change is due to the application of linear models combined with that the operation point is moved, and thus also the trust-regions. As a consequence, finding the gradient of the Lagrangian at the current operation point is (close to) impossible.

If the norm of the gradient of the Lagrangian was evaluated at the solution of (8) before the operation point is moved, then this point satisfies the FONC, and the multipliers are available. Despite this not being the gradient at the operation point, it will be used as the measure of stationarity. If the norm of this gradient is small, it indicates that the gradient's elements are close to zero without considering the trust-region bounds. And in this case, we are indeed close to a stationary point (of the model) and a reduction of the trust-region radii are in-order to "zoom in" on the true function.

Another complicating factor of implementing the criticality step for the constrained case is a result of the previous issues. The criticality step in Conn et al. (2009) is an iterative procedure which terminates when there is a satisfactory relationship between the norm of the gradient and the trust-region radius: $\Delta \leq \mu_c \|g\|$, where g is the model gradient and μ_c is a tuning parameter. This iterative procedure will be exited as long as the norm of the gradient of the true function at the center point is not zero. Roughly explained, each iteration of the criticality step consists of (i) reducing the radius ($\tilde{\Delta} \leftarrow \omega_c \tilde{\Delta}, \omega_c \in (0,1)$), (ii) making sure the (new) set of points are poised in the new region, (iii) creating the new surrogate model with gradient $\tilde{g}$, and finally (iv) check if its gradient is satisfactory. When the procedure exits, the final radius is chosen as

$$\Delta = \min(\max(\tilde{\Delta}, \beta_c \|\tilde{g}\|), \Delta_0) \tag{14}$$

where Δ_0 is the radius when entering the criticality step. The constants must satisfy $\mu_c > \beta_c > 0$. The final radius Δ of the criticality step is selected as the radius in the range $[\tilde{\Delta}, \Delta_0]$ closest to $\beta_c \|\tilde{g}\|$, which helps avoiding that the radius is reduced too much.

Such an iterative procedure is not feasible for our setup due to the first issues. The infeasibility will be illustrated through an example. Assume that the radius was reduced because the gradient was found too small compared to the radius: $\tilde{\Delta} \leftarrow \omega_c \tilde{\Delta}, \omega_c \in (0,1)$. Next step would be to create (potentially new) sets of points that are deemed poised. Further, the surrogate models are made, the subproblem is solved and the true function is evaluated at the solution. Now, if the prediction quality was not satisfied for all the considered units, then the gradient of the Lagrangian could not be evaluated. Without this gradient, the criticality step cannot continue, and thus the iterative setup of the criticality step in Conn et al. (2009) is not feasible.

To deal with this, we suggest a single pass approach which tries to copy the most important feature of the original criticality step in Conn et al. (2009). More specifically, the radii are reduced if at least one of the elements of the gradient is too small compared to the corresponding radius. The proposed criticality step consists of only one pass, and is hereafter referred to it as the criticality *substep*. The substep is given in Algorithm 1 .

This criticality substep will result in the radii (potentially) decreasing faster towards zero, and maybe even too fast. As opposed to the trust-region selection in (14), we simply get $\Delta = \tilde{\Delta} = \omega_c \Delta_0$ which would correspond to setting $\beta_c = 0$. As alluded to in (10), we have introduced a $\Delta_{\min}$, which also will be used in the criticality step, see Algorithm 1. This hard limit will avoid that the algorithm gets stuck due to the trust-region radii going too fast to zero.

Algorithm 1 Criticality substep

```
Parameters: $\mu_c > 0$, $0 < \omega_c < 1$, $\mathbf{\Delta}_{\min}$
1: procedure CRITICALITY($\mathbf{\Delta}$, $\mathcal{U}$, $\mathbf{z}_{\mathrm{op}}$, $\mathbf{u}^+$, $\mathbf{z}^+$, $\lambda^+$, $\underline{\mu}^+$, $\overline{\mu}^+$)
2:     Require: All $U_i$'s are deemed poised for interpolation (or regression).
3:     Calculate $\nabla_{\mathbf{u}}\mathcal{L}$ in (12) at the solution.
4:     reduce = False
5:     for $i = 1, 2, \dots, n_u$ do
6:         if $z_i^+ == 1$ and $\mu_c \left|\nabla_{\mathbf{u}_i}\mathcal{L}\right| < \Delta_i$ then
7:             reduce = True
8:         end if
9:     end for
10:    if reduce then
11:        for $i = 1, 2, \dots, n_u$ do
12:            if $z_i^+ == 1$ and $\mu_c \left|\nabla_{\mathbf{u}_i}\mathcal{L}\right| < \Delta_i$ then
13:                $\Delta_i \leftarrow \max(\omega_c \Delta_i, \Delta_{i,\min})$
14:            end if
15:        end for
16:    end if
17:    return $\mathbf{\Delta}$
18: end procedure
```

The setup of having several units, trust-region radii and models allows for some design choices. Specifically, if one element of the gradient is zero, the corresponding unit's radius or all radii could be decreased. The latter has shown to result in better performance. This is further illustrated and discussed in Section 5.2. This choice is not without its challenges. The more aggressive reduction of the radii may indeed hinder fast convergence. However, this may also be the case for either approach, and will be both problem specific and rely on the chosen parameters. For example, starting with small trust-region radii, and/or having strong contraction parameters (*i.e.*, $\omega_c \approx 0$ and $\gamma \approx 0$), will all result in the radii hitting the $\Delta_{\min}$ quickly.

2.7. The complete algorithm

All the required building blocks of the algorithm have been detailed in the previous sections. The only extension that is missing is how to exploit the unused degrees of freedom while finding geometry improvement points. The presentation is postponed to after this section. The complete derivative-free trust-region model-based approach is given in Algorithm 2. A flowchart of the algorithm is shown in Fig. 2.

The algorithm is inspired by the framework presented in Conn et al. (2009), and specifically Algorithm 10.1. They differ in the following aspects.

- First, the way poisedness is checked and improved is different. In this paper, emphasis is put on not disturbing the to-be-optimized process more than necessary when poisedness needs to be improved, see Section 3. Such considerations are not made in Conn et al. (2009).
- Second, the constrained optimization problem (1) introduces the need of looking at the gradient of the Lagrangian instead of the objective function in the criticality step.
- Third, the integer variables z_i's introduce additional design choices.
- Fourth, each unit is modeled individually with its own trust-region radius, whereas they use only one model.
- Fifth, due to the poisedness being checked and improved in the beginning of the while loop in Algorithm 2, the remaining part of the algorithm concerning the poisedness property is simplified compared to Algorithm 10.1 in Conn et al. (2009).
- Sixth, all radii will be reduced if (i) the solution of the subproblem was not better than the current $\mathbf{u}_{\mathrm{op}}$, and (ii) none of the radii has been reduced during the current iteration of the algorithm. This ensures that the radii will decrease towards $\Delta_{\min}$ whenever the algorithm may get stuck. More explicitly, if all sets are deemed

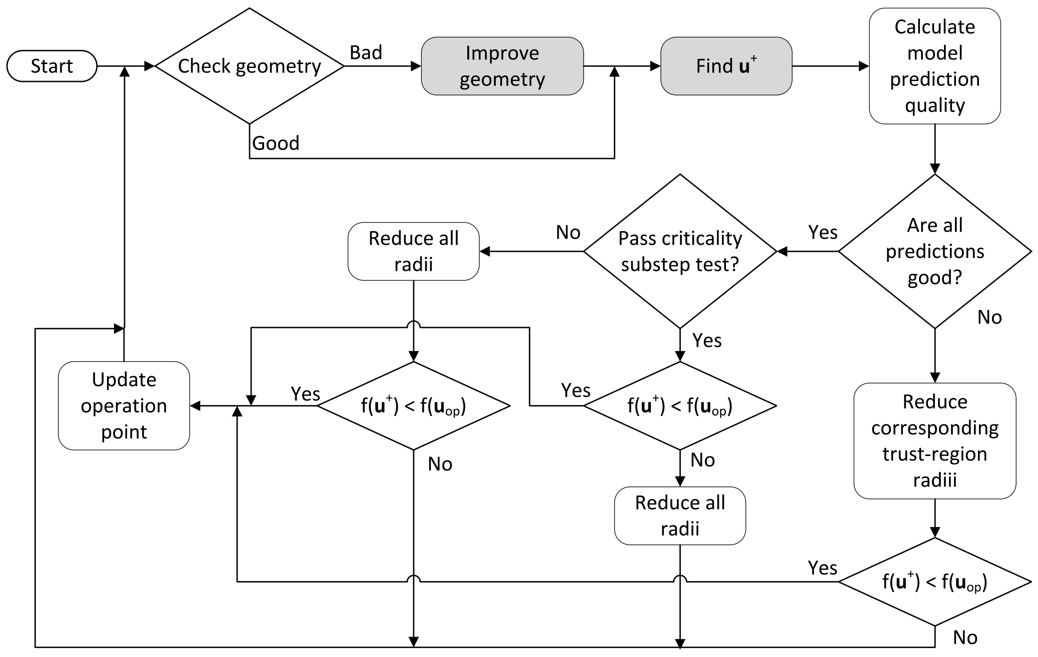

Fig. 2. A visual flowchart for Algorithm 2. The gray boxes are where the true function is evaluated.

poised, all model prediction qualities satisfy the criterion, and the norm of the gradient of the Lagrangian is sufficiently large, but the solution of the subproblem is worse (or equally good) as the one at $\mathbf{u}_{\text{op}}$, then all the radii are reduced.

Deciding a termination criterion that consistently provides both a good final solution and avoiding unnecessarily many perturbation is not straightforward. The easiest criterion is to stop after a certain amount of true function evaluations. This could be a valid choice if the user decides that a certain amount of time should be used for improving the performance. Then by knowing how long it takes to perform one true function evaluation, one may calculate how many evaluations are available. Another option could be to track the relative improvement of the solutions of (8). Then, once the trust-region radii (for the considered units) are all at their corresponding minima, one could terminate once the experienced improvement is below some threshold.

3. Geometry improvement: Unused degrees of freedom

The optimization formulation in (7), assuming it is feasible, provides a geometry improvement point for all the units that require such a point. Depending upon how many units need a new point, there might be several degrees of freedom left in the optimization. Put differently, there are may be many different $\mathbf{u}$ vectors that will satisfy the geometry requirement. This freedom will be exploited next.

We emphasize that nothing of what is added from here on out will actually restrict the feasible area. In other words, if (7) is feasible, so is the to-be-presented problem. The extensions suggested next could be altered to fit the specific application.

3.1. Minimize change

When performing a geometry improvement step, it could be tempting to keep the objective function as in (1a), but where the f_i's were replaced by their linearized counterpart. This way, the true function would be (attempted) minimized while obtaining the required points. However, it is important to remember that the reason new points are required is because the unit models are not to be trusted.

Before presenting the suggested objective function, we highlight the difference between operation point and center point. The operation point and the center point for a unit will be the same if the unit is on. If a unit is off, its operation point is 0, but its center point will be kept as the last non-zero operation point.

Algorithm 2 Derivative-free trust-region optimization method for resource allocation problems

```
Input: Operation point $(\mathbf{u}_{\text{op}}, \mathbf{z}_{\text{op}})$, and center-point $\mathbf{u}_{\text{cp}}$. Trust-region radii parameters $\mathbf{\Delta}$, $\mathbf{\Delta}_{\text{min}}$, and $r$. Trust-region radii update parameters $0 < \gamma < 1$ and $0 < \eta_1 < 1$. Criticality step parameters $\mu_c > 0$ and $0 < \omega_c < 1$
Require: $\mathbf{u}_{\text{op}}$ is feasible
Require: Either all units have a model, or all units are on $z_i = 1$
 1: while not converged do
 2:     $\mathbf{\Delta}_p \leftarrow \mathbf{\Delta}$
 3:     Check if all $\mathcal{U}_i$'s are deemed poised according to definition in Section 2.1.1 (or 2.1.2)
 4:     If any units need more points, solve e.g., (7) or (32), and evaluate the true function
 5:     Find $(\mathbf{u}^+, \mathbf{z}^+, \lambda^+, \underline{\mu}^+, \overline{\mu}^+)$ by solving the subproblem (8)
 6:     Evaluate the true function
 7:     Calculate all considered units' $\rho_i$'s in (9)
 8:     if All calculated $\rho_i$'s $\geq \eta_1$ then
 9:         $\mathbf{\Delta} = \textsc{Criticality}(\mathbf{\Delta}, \mathcal{U}, \mathbf{z}_{\text{op}}, \mathbf{u}^+, \mathbf{z}^+, \lambda^+, \underline{\mu}^+, \overline{\mu}^+)$
10:     else
11:         for $i = 1, 2, \ldots, n_u$ do
12:             if $\rho_i < \eta_1$ then
13:                 $\Delta_i \leftarrow \max(\gamma \Delta_i, \Delta_{i,\text{min}})$
14:             end if
15:         end for
16:     end if
17:     if $f(\mathbf{u}^+) < f(\mathbf{u}_{\text{op}})$ then
18:         $\mathbf{u}_{\text{op}} \leftarrow \mathbf{u}^+$
19:         $\mathbf{z}_{\text{op}} \leftarrow \mathbf{z}^+$
20:     else
21:         if $\mathbf{\Delta} == \mathbf{\Delta}_p$ then
22:             for $i = 1, 2, \ldots, n_u$ do
23:                 $\Delta_i \leftarrow \max(\gamma \Delta_i, \Delta_{i,\text{min}})$
24:             end for
25:         end if
26:     end if
27:     Add $\mathbf{u}^+$ to the set $\mathcal{U}_{\text{all}}$.
28: end while
```

We suggest to minimize the normalized squared distance from the operation point:

$$P_u \sum_{i\in\mathcal{Z}_{\text{on}}} \left(\frac{u_i - u_{i,\text{op}}}{\overline{u}_i - \underline{u}_i}\right)^2 \tag{15}$$

where $\mathcal{Z}_{\text{on}}$ is the set of indices of the units that are on at the current operation point, and P_u is a scaling parameter used for prioritizing different objectives later on. A squared distance would be indifferent to u_i increasing or decreasing. However, there could be available information that would give preference to either increase or decrease.

3.2. Priority to increase/decrease

The following extension allows for prioritized exploration directions for both the lacking units and the neutral units. Let k be an index of a neutral unit, *i.e.*, one that does not need a new point. The binary variable $x_{k,l}$ will be 1 if $u_k \le u_{k,\text{cp}}$ and $x_{k,r} = 1$ if $u_k \ge u_{k,\text{cp}}$. This is similar to the use of the $x_{j,l}$'s and $x_{j,r}$'s in (7). The constraints below must be added to (7)

$$u_k \ge u_{k,\text{cp}} x_{k,r} + \underline{u}_k x_{k,l} \tag{16a}$$

$$u_k \le u_{k,\text{cp}} x_{k,l} + \overline{u}_k x_{k,r} \tag{16b}$$

$$z_k = x_{k,l} + x_{k,r} \tag{16c}$$

In addition, the objective function should be extended with:

$$P_{l,r} \sum_{i=1}^{n_u} x_{i,l} \cdot v_{i,l} + x_{i,r} \cdot v_{i,r} \tag{17}$$

where the $v_{i,l}$'s and $v_{i,r}$'s are the scalars giving priority to changing u_i in either direction:

$$v_{i,l} < 0 \text{ and } v_{i,r} = 0 \text{ if prioritize to decrease } u_i \tag{18}$$

$$v_{i,l} = 0 \text{ and } v_{i,r} = 0 \text{ if no preference} \tag{19}$$

$$v_{i,l} = 0 \text{ and } v_{i,r} < 0 \text{ if prioritize to increase } u_i \tag{20}$$

The $v_{i,l}$'s and $v_{i,r}$'s for the neutral units could for example be chosen as:

$$v_{k,l} = -1, v_{k,r} = 0 \quad \text{if } |a_k| \le 10^{-4} \text{ or } a_k > 0 \tag{21a}$$

$$v_{k,l} = 0, v_{k,r} = -1 \quad \text{else} \tag{21b}$$

where a_k refers to the slope parameter in the linear model Eq. (2). For the lacking units, it is less obvious how to choose $v_{j,l}$'s and $v_{j,r}$'s. If no knowledge is available, set them to 0.

With these two extensions, it will be prioritized to stay as close as possible to the operation point and the exploration directions can be prioritized. Next up, an extension to prioritize which neutral unit to pertub will be presented.

3.3. Prioritize which neutral unit to alter

When some units need additional points to make a good linear model, other units must be perturbed too. Ideally, the resource should be distributed only between the units that need changes, but such redistribution of the resource may not always be sufficient. There may be some units that one consider as prioritized as they have an low associated cost, and these should not be altered unless it is necessary. This mindset is perhaps more intuitive in the maximization setting where one would not like to alter units that gives a high revenue.

A new set of binary variables will be used to impose this prioritization. Let w_i be a binary variable to indicate if a unit i is allowed ($w_i = 1$) to be altered or not ($w_i = 0$).

$$u_i \le (z_i - w_i)u_{i,\text{op}} + w_i \overline{u}_i \tag{22}$$

$$u_i \ge (z_i - w_i)u_{i,\text{op}} + w_i \underline{u}_i \tag{23}$$

$$w_i \le z_i \tag{24}$$

with $i \in \mathcal{Z}_{\text{on}}$, and where subscript "op" refers to the current operation point. The equations above will only be imposed for units that are currently on ($z_{i,\text{op}} = 1$). Turning off a unit ($z_i = 0$) does not count as altering, and such changes will be handles later.

The objective function then needs to be extended with

$$P_w \sum_{i\in\mathcal{Z}_{\text{on}}} w_i W_i \tag{25}$$

and $\mathbf{W} = [W_1, W_2, \dots, W_{n_u}]^\top$ is a vector with non-negative elements and contains the prioritization weight for perturbing the units. A lower value means it is more acceptable to perturb it. A value of 0 should be given to the lacking units.

Deciding the remaining W_i's is up to the user of the method. It could be based on application/engineering knowledge, or some kind of measurement of "importance". One idea could be to take an inverted average over all previously gathered points ($\mathcal{U}_{i,\text{all}}$), including those outside the trust-region for the neutral units:

$$W_{i,\text{avg}} = \frac{1}{n_i} \sum_{j=1}^{n_i} \frac{1}{\mathcal{F}_{i,\text{all}}[j]} \tag{26}$$

where $\mathcal{F}_{i,\text{all}}$ is an ordered set containing the measured value $f_i(u_{i,j})$ for each element $u_{i,j} \in \mathcal{U}_{i,\text{all}}$. Further, the $\mathbf{W}_{i,\text{avg}}$ should be shifted and scaled to obtain non-negative elements:

$$W_{i,\text{shifted}} \leftarrow W_{i,\text{avg}} + |\min(0, \mathbf{W}_{\text{avg}})| + 1 \tag{27}$$

$$W_i \leftarrow \frac{W_{i,\text{shifted}}}{\max(\mathbf{W}_{\text{shifted}}) - \min(\mathbf{W}_{\text{shifted}})} \tag{28}$$

where the +1 is added to ensure that the weight for the neutral units is higher than 0, $W_k > 0$, and thus more costly to alter than the lacking units ($W_j = 0$).

One last set of constraints is added to (7) to obtain the desired prioritized altering. Let $\mathcal{P}$ be an ordered set of n_u indices of the units ordered according to an ascending importance. *I.e.*, it is most acceptable to perturb the unit whose index is found at first element of $\mathcal{P}$. Further, let $\mathcal{M}$ be a one-to-one mapping from unit i to its corresponding index in $\mathcal{P}$. The inverse mapping (from priority index to unit index) is given as $\mathcal{M}^{-1}$

$$w_{\mathcal{M}^{-1}(n_u)} \le w_{\mathcal{M}^{-1}(n_u-1)} \le \dots \le w_{\mathcal{M}^{-1}(1)} \tag{29}$$

3.4. Prioritize to not change on/off status

Depending upon the application area, it may be costly and/or undesirable to turn on/off units simply to get geometry improvement points. To penalize closing a unit, the following can be added to the objective function:

$$P_z \sum_{i\in\mathcal{Z}_{\text{on}}} -z_i \tag{30}$$

For the units which are currently off, a similar penalization scheme can be used. However, a preferred turning on order can be imposed by adding the single extension:

$$\sum_{i\in\mathcal{Z}_{\text{off}}} (P_z + P_w(1 - W_i))z_i \tag{31}$$

where $(1 - W_i)$ is used because a value of W_i closer to one indicates it is a unit which is believed to have a low associated cost.

3.5. Geometry improvement — summary

An optimization problem with all the aforementioned extensions is now presented.

$$\min_{\mathbf{u},\mathbf{z},\mathbf{x},\mathbf{w}} \quad P_u \sum_{i\in\mathcal{Z}_{\text{on}}} \left(\frac{u_i - u_{i,\text{op}}}{\overline{u}_i - \underline{u}_i}\right)^2$$

$$+ P_{l,r} \sum_{i=1}^{n_u} x_{i,l} v_{i,l} + x_{i,r} v_{i,r}$$

$$+ \sum_{i \in \mathcal{Z}_{\text{on}}} \left(P_w w_i W_i - P_z z_i \right)$$

$$+ \sum_{i \in \mathcal{Z}_{\text{off}}} (P_z + P_w (1 - W_i)) z_i \tag{32a}$$

$$\text{s.t.} \quad \sum_{i=1}^{n_u} u_i = u_{\max} \tag{32b}$$

$$\underline{u}_i z_i \le u_i \le \bar{u}_i z_i \tag{32c}$$

$$u_j \ge \underline{u}_j z_j + (u_{j,\text{cp}} - \Delta_j - \underline{u}_j) x_{j,l}$$

$$+ (u_{j,\text{cp}} + 0.5\Delta_j + \Delta_{=} - \underline{u}_j) x_{j,r} \tag{32d}$$

$$u_j \le \bar{u}_j z_j + (u_{j,\text{cp}} + \Delta_j - \bar{u}_j) x_{j,r}$$

$$+ (u_{j,\text{cp}} - 0.5\Delta_j - \Delta_{=} - \bar{u}_j) x_{j,l} \tag{32e}$$

$$x_{j,l} + x_{j,r} = z_j \tag{32f}$$

$$u_k \ge u_{k,\text{cp}} x_{k,r} + \underline{u}_k x_{k,l} \tag{32g}$$

$$u_k \le u_{k,\text{cp}} x_{k,l} + \bar{u}_k x_{k,r} \tag{32h}$$

$$x_{k,l} + x_{k,r} = z_k \tag{32i}$$

$$w_{\mathcal{M}^{-1}(n_u)} \le w_{\mathcal{M}^{-1}(n_u-1)}$$

$$\le \dots \le w_{\mathcal{M}^{-1}(1)} \tag{32j}$$

$$u_i \ge w_i \underline{u}_i + (z_i - w_i) u_{i,\text{op}}, \quad \forall i \in \mathcal{Z}_{\text{on}} \tag{32k}$$

$$u_i \le w_i \bar{u}_i + (z_i - w_i) u_{i,\text{op}}, \quad \forall i \in \mathcal{Z}_{\text{on}} \tag{32l}$$

$$w_i \le z_i, \quad \forall i \in \mathcal{Z}_{\text{on}} \tag{32m}$$

$$u_i \in \mathbb{R} \tag{32n}$$

$$z_j = 1 \tag{32o}$$

$$z_i \in \{0,1\} \quad \text{On/Off} \tag{32p}$$

$$x_{i,l}, x_{i,r} \in \{0,1\} \quad \text{Preferred perturbation direction} \tag{32q}$$

$$w_i \in \{0,1\} \quad \text{Preferred perturbation order} \tag{32r}$$

where indices i, j and k indicate it concerns all units, lacking units, and neutral units, respectively, unless otherwise explicitly stated. In contrast to (7), which was a MILP, the new geometry improvement problem is a Mixed-Integer Quadratic Program.

The scaling parameters, or prioritization parameters, P's, in (32a) can be used to give priority to the different extensions. The prioritization order could depend upon the application area. We suggest to use the following decreasing order of importance:

1. Do not alter the on/off status (unless necessary).
2. Alter the neutral units in (inverse) order of importance.
3. The perturbation direction.
4. Stay as close as possible to the current operation point

To obtain this behavior, the P's are selected as follows. Both P_u and $P_{l,r}$ are set to 1. Considering that the first term of (32a) is indifferent to the direction, it is not in conflict with the second term. We assume that the v's are selected according to (21), which ensures that the two first terms for a single unit is in the range $[-1, 0]$. P_w must be selected large enough to ensure that perturbing left/right will not be more advantageous than not perturbing at all, $P_w = n_u + 1$. Finally, P_z must be sufficiently large such that opening/closing *one* unit cannot lead to an improved objective function, $P_z = n_u(n_u + 1)$.

4. Application study — gas lift allocation

This section starts by first explaining the motivational example of this work. Thereafter, a decomposition of a specific curve which allows for utilization of the available sensor data will be introduced. To the best of the authors' knowledge, this decomposition has not previously been used to exploit data in such a manner. The suggested algorithm will be applied to a small simulated version of the motivational example. Problems of larger sizes will be tackled in the next section.

This work was motivated by a part of the Daily Production Optimization (DPO) challenge in the oil and gas industry. The DPO is concerned with optimally utilizing the available resources while still obeying system and operational constraints. The study consists of a set of wells producing to a separator. The wells produce fluids comprised of oil, gas and water.

Initially, the pressure inside the reservoir below the seabed is, typically, sufficiently high to drive the flow of hydrocarbons to the surface on its own. As fluids are drained, the reservoir pressure decreases and at some point it is too low to maintain an economically beneficial production.

One method to increase production when the reservoir pressure has decreased, is to inject gas into the wellbore. The gas, or lift gas, is mixed with the fluids coming from the reservoir, and this mixture will achieve a lower density, and, thus, the production of fluids will increase. However, there is a limit on how much gas may be injected before the production actually starts to decrease. This is due to that the friction of the mixture will increase faster than the mass of the mixture decreases.

Each well may have a limitation on the lower bound on the gas lift injection rate. A well may exhibit unstable behavior, *e.g.* slugging, if too little gas is injected. In addition, an upper limit may exist. A well may produce sand (from the reservoir) if the pressure difference between the reservoir and the wellbore is too large. In this work, these constraints are given as bounds on the gas lift injection rates.

Distributing the available lift gas between the wells can be seen as a resource allocation problem on the form (1). The resource, or lift gas, must be allocated between the wells, or units. In addition, the natural choice of opening or shutting wells is considered.

A standard method of approaching this problem is to create a Gas Lift Performance Curve (GLPC) for each well. This concave curve gives the relationship between the gas lift rate and the oil production in steady-state conditions for the well. If these curves were available and reliable, they should be used and the resource allocation problem is straight forward. However, the focus of this example is when they are not available.

The gatherable data for each well can be split into two categories; production data and test-separator data. The production data are collected from sensors attached to that specific well and can be read any time. For our setup, that includes the downhole pressure gauge and the gas lift rate sensor. There is typically no rate sensor attached to the output of each well which is why the second set of data is acquired. The test-separator data is only available when the considered well is routed to a test-separator. For our setup, the relevant data is the oil rate.

An offshore oil field typically has many wells, but only one test-separator. In addition, letting a well produce to the test-separator invariably implies a cost of lost production. Therefore, test-separator data may be old, and only a few steady-state measuring points are taken. Also, in many fields well production may be off-design with respect to test-separator instrumentation, which implies significant uncertainties in test-separator measurements. Taken together, these issues imply that getting more than first-order (linear) information from test-separator data is a challenge.

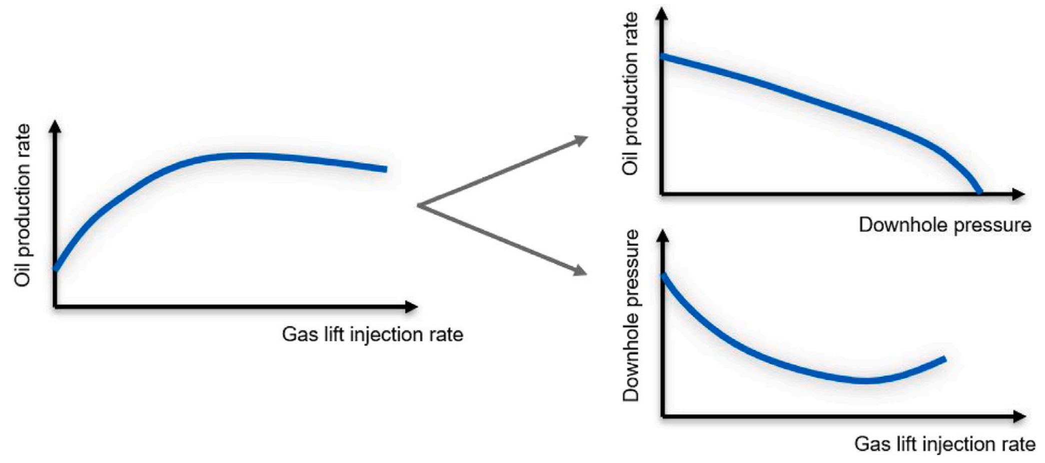

Fig. 3. Illustration of the suggested splitting of the GLPC.

4.1. *Gas lift performance curve decomposition*

We suggest to decompose the GLPC into two curves. One curve relates the gas lift rate to the downhole pressure, and another which relates the downhole pressure to the oil rate, see the illustration in Fig. 3.

This decomposition allows for an exploitation of the two different categories of data. The first curve will only be updated based on data collected from the test-separator analysis. As mentioned above, due to the challenges with the test-separator measurements, this first curve will be linear. In addition, the true relationship between the downhole pressure and the oil rate is typically close to linear for a large range of pressures.

The real advantage of this decomposition surfaces when the second curve is considered. This curve, which relates the gas lift rate to the downhole pressure, can be updated based on data collected during production as both these measurements are available. The relationship between gas lift rate and downhole pressure may better be modeled by a quadratic equation. However, the extrapolation capabilities of a quadratic model, or any polynomials of degree 2 or higher, may be unreliable. Thus, a linear model is chosen for both curves. To deal with the inaccuracy of the assumed linear relationship between gas lift rate and downhole pressure, the presented trust-region framework will be applied.

4.2. *Setup information*

This section contains details on the setup and parameters. The setup given here is used in the subsequent examples unless otherwise stated.

The true linear relationship between the downhole pressure and the oil rate for each well is assumed measured perfectly. Introducing significant errors in these measurements would make for a more realistic case. However, the proposed algorithm will only find good solutions to the extent that the given set of relationships reflects reality. Moreover, the comparison of the results of the algorithm with the optimal solution becomes clearer when these relationships are modeled correctly. It is not expected that the algorithm will converge to a local solution if the downhole pressures versus the oil rates are modeled incorrectly. These relationships can be thought of some kind of ranking between the different wells.

In order to focus on the core of the algorithm, and not on *e.g.* filtering, no noise is introduced in the measurements. The modeled linear relationship between the gas lift rate and the downhole pressure for each well is found using the linear least squares regression method `lsq_linear` in Python's Scipy. Only two points for each well are used to create the models: center point plus one more.

Table 1
The default parameters used for the algorithm.

Parameter	Value
η_1	0.1
γ	0.7
ω_c	0.7
μ_c	10^9
$\mathbf{\Delta}_0$	2000
$\mathbf{\Delta}_=$	0.0
$\mathbf{\Delta}_{\min}$	50.0
r	1.0

The parameters used in the algorithm are given in Table 1.

The optimization problems (8) and (32) are formulated using CasADi (Andersson et al., 2019), and solved with the Mixed-Integer Nonlinear Program (MINLP) solver BONMIN (Bonami et al., 2008).[1] The computational study was carried out on Dell XPS 15 9570 with Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz.

The algorithm is initialized with, or started from, the current operation point, which is the natural choice. The available lift gas throughout subsequent iterations of the algorithm is given as the sum of the applied gas lift rates at the initial operation point. Experience shows that solving the geometry improvement optimization problem (32) with many units requiring new points can be a time consuming task. Applying a better solver and/or take advantage of high performance computing could ease this task. However, to circumvent this waiting time, a limit on the amount of lacking units for each time (32) is solved is imposed and set to 10. If, for any reason, the solver cannot find a solution to (32), then that set of lacking units is split in two and the formulation is solved twice with the easier problem. This splitting may happen as many times as is required.

Throughout the entire application study the curves and points referred to as "optimal" or "optimum" are found by using BONMIN on the problem with all the correct Gas Lift Performance Curves (GLPCs) available. *I.e.*, the underlying problem is formulated as in (1) with the

[1] Unfortunately, the Lagrange multipliers required for evaluating the gradient (12) is not available directly from BONMIN. To overcome this, the subproblem (8) is first solved with BONMIN, then it is solved again with the Nonlinear Program (NLP) solver IPOPT (Wächter and Biegler, 2006) keeping the binary variables fixed.

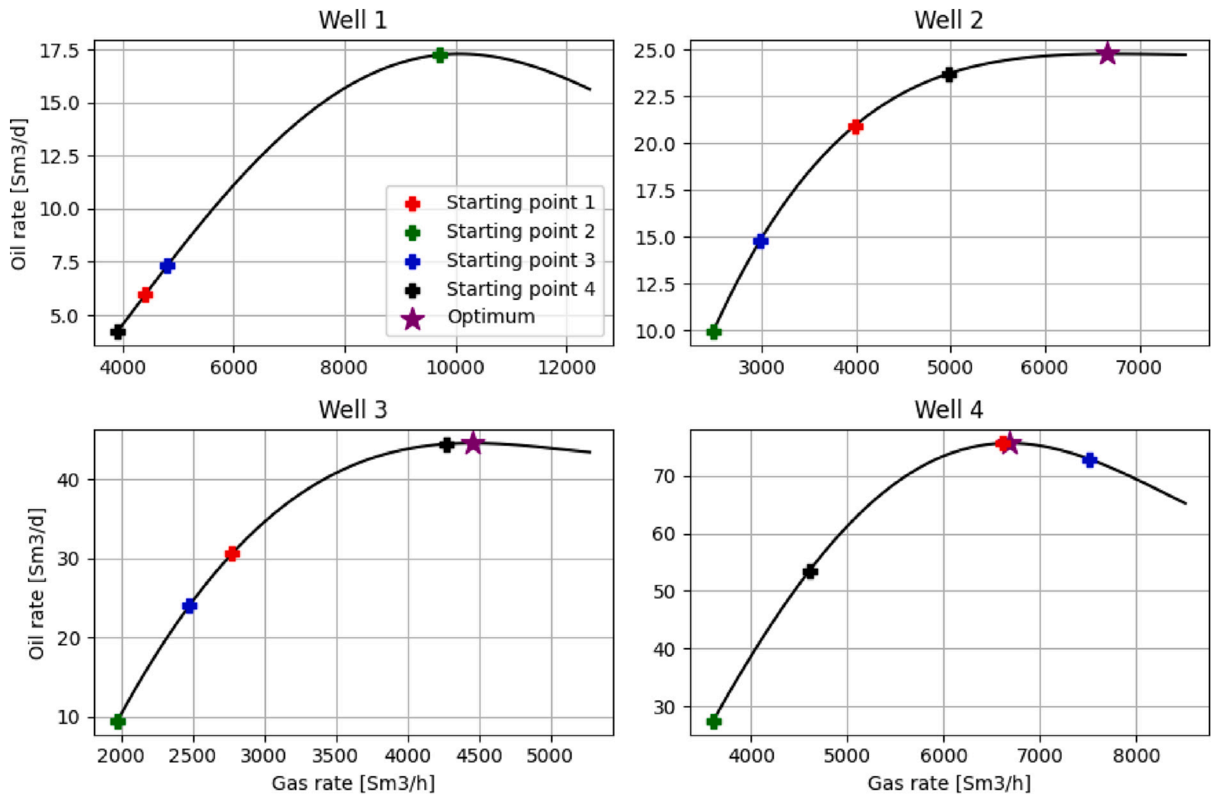

Fig. 4. The GLPCs for the four wells case study in Section 4.3. Notice that Well 1 is shut at the optimum and therefore does not have a star marker. Each set of equally colored plus symbols shows the start point for one case.

f_i's available, and solved by an off-the-shelf solver. This is in contrast to the proposed algorithm, which was designed to only have noisy samples of the GLPCs (the f_i's) available.

The termination criterion of Algorithm 2 is set as a threshold on true function evaluations. At the end of each iteration of the algorithm, it checks if the true function has been evaluated equally or more times than this threshold.

4.2.1. *Consequences of using two linear models*

As a result of decomposing the gas lift performance curve into two curves, each unit will have two linear models associated to it. One which relates the gas lift rate to the oil rate (through the downhole pressure),

$$Q_i(u_i) = \alpha_i P_i(u_i) + \beta_i \tag{33}$$

where $P_i(u_i)$ is the second linear model relating the gas lift rate to the downhole pressure

$$P_i(u_i) = a_i u_i + b_i \tag{34}$$

The parameters of (33) remains fixed, whereas those in (34) are updated by the algorithm. In the theory section, only one linear model where used. The tuple (u_i, Q_i) is used in most steps, except when the model prediction quality is calculated in (9). The model prediction quality requires a comparison between the predicted and measured values, thus, the tuple (u_i, P_i) is used for this.

The algorithm is presented as a minimization method. The goal is to maximize the (estimated) oil rate, thus, (33) must be multiplied by negative one in the objective function (8) such that the oil rate is maximized and not minimized.

4.3. *Results for a four well case*

A small example consisting of four wells, or units, is presented. This allows for easier illustration of how each well behaves. In this four well example, we investigate how the algorithm evolves as a function of the number of true function evaluations. Four different starting operation points are considered. For all starting points, the total available lift gas are the same.

The four GLPCs and the four sets of starting points are shown in Fig. 4. For example, the starting points for Starting point 1 may be found as all the red marks in the four subplots. The optimum is the same for all different initial conditions. The lower and upper bounds for the wells are available in Fig. 4 as the minimum and maximum x-values for the curves. There is no marker for the optimum for Well 1 as this well is shut at the optimum.

It is emphasized that the concave GLPCs are unknown to the optimizer and they may only be evaluated.

The evolution of the total oil rate, gas lift rates, and trust-region radii are shown in Fig. 5, Fig. 6, and Fig. 7, respectively. In almost all cases, the trust-region radii have reached the lower limit $\Delta_{\min}$ for the wells that are open, see Fig. 7.

In Fig. 5, it can be seen that the algorithm has more or less reached the (local) optimum in less than 15 evaluations of the true functions. For Starting point 1, 3 and 4, the optimum found by the algorithm is the same as the one found by BONMIN. For Starting point 2, we can see that the algorithm does not reach the optimum. Looking at Fig. 6, it can be seen that the algorithm converges to a point where one of the wells are closed. The proposed algorithm is not a global algorithm, and converging to a local optimum is expected behavior. Nonetheless, for the given set of open wells, the found solution is indeed a local optimum. The dotted line in Fig. 5 is found by BONMIN when the binary variables are fixed. In this case, the solutions found by BONMIN and the algorithm coincide.

For all the different starting points, no wells were ever closed for the reason of improving the geometry of other wells.

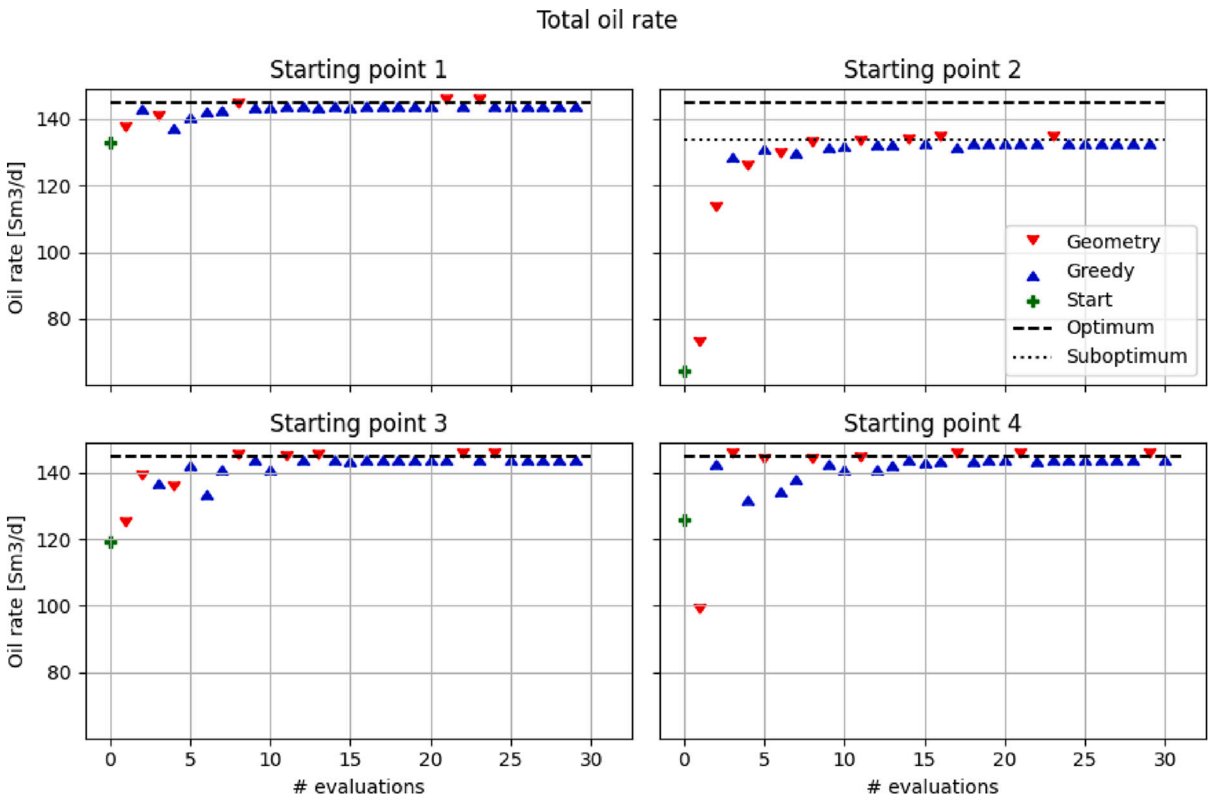

Fig. 5. The evolution of the total oil rate for the four wells case study in Section 4.3. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry.

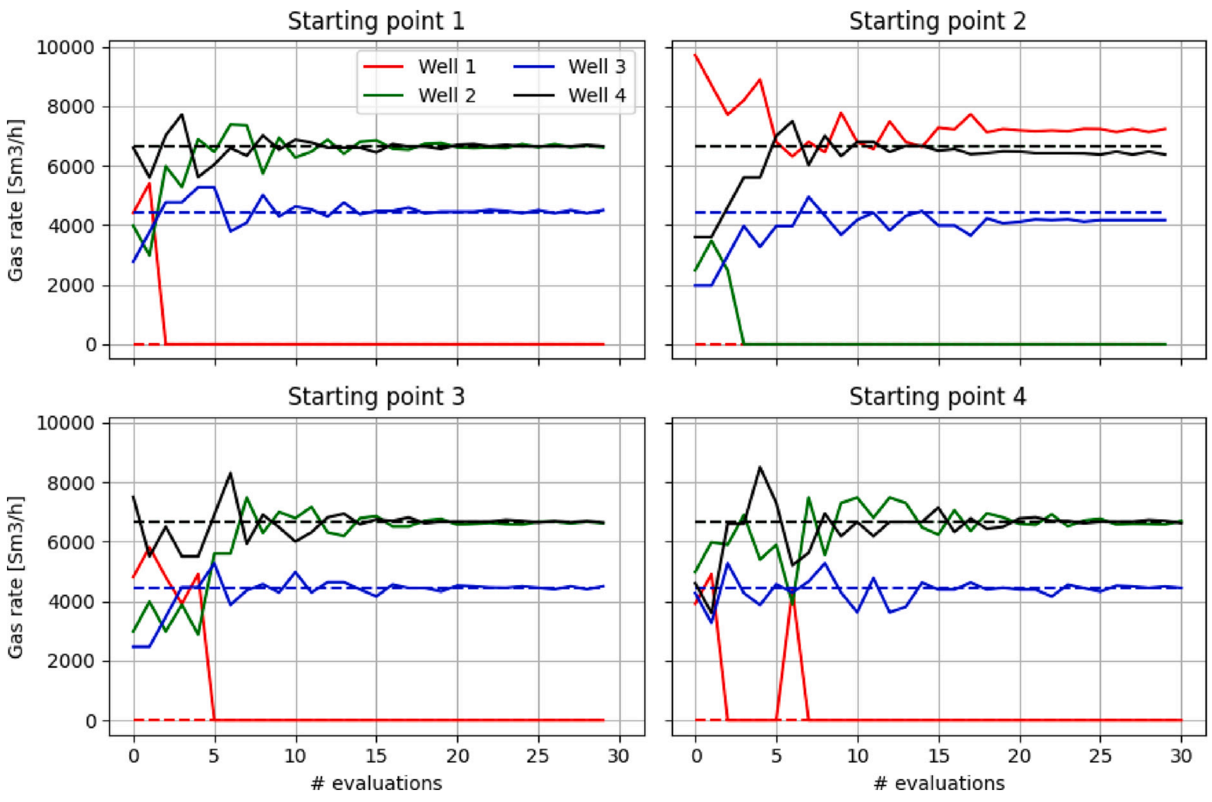

Fig. 6. The evolution of the gas lift rates for the four wells case study in Section 4.3.

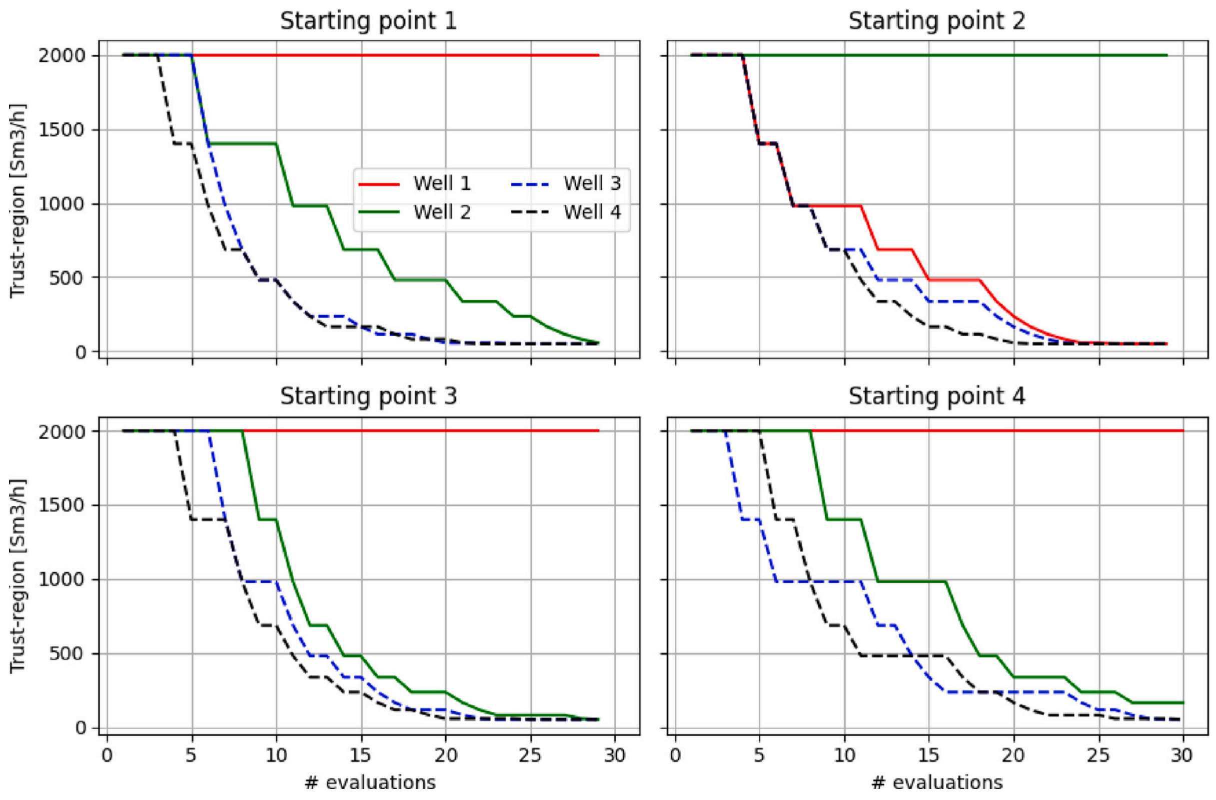

Fig. 7. The evolution of the trust-region radii for the four wells case study in Section 4.3.

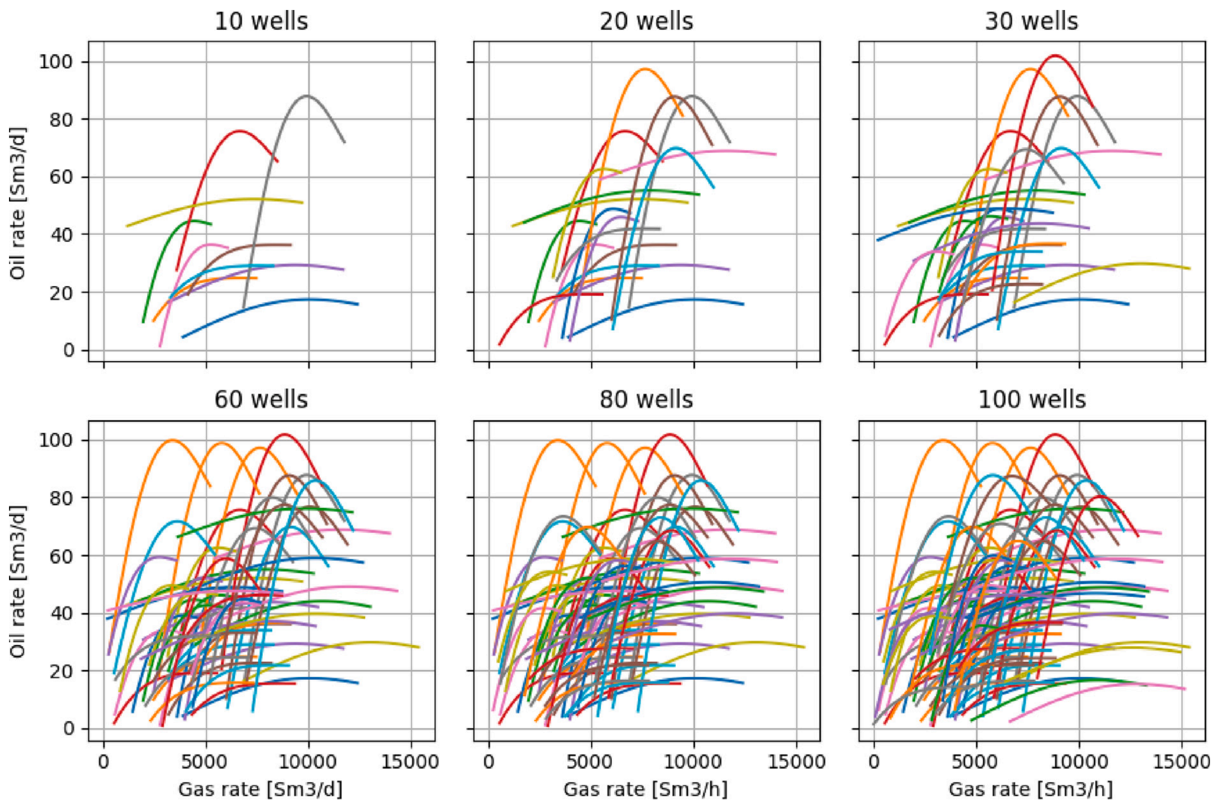

Fig. 8. The GLPCs for all the wells used in Section 5.1.

Table 2
Amount of available lift gas for the scenarios with an increasing number of wells in Section 5.1. All numbers are in Sm3/h.

	N = 10	N = 20	N = 30	N = 60
Starting point 1	33524	72172	105982	200805
Starting point 2	43524	92172	135982	260805
Starting point 3	61974	126422	188682	363555
	N = 80	N = 100		
Starting point 1	1268271	335670		
Starting point 2	2348271	435670		
Starting point 3	3485271	606920		

5. Investigating properties of the algorithm

In this section, different aspects of the algorithm will be analyzed. The examples in this section will be different instantiations of the motivational example presented in the previous section.

This section is structured as follows. First, the algorithm is applied to several setups with an increasing number of wells to see how the algorithm performs. Second, an algorithmic design choice made in the theory section is justified through examples. Third, the importance of the minimum trust-region radius $\Delta_{\min}$ parameter is illustrated by examples. Fourth, parameters of the algorithm related to the radii are discussed. Fifth, the importance of the starting point, for a certain amount of lift gas, is investigated.

5.1. Complexity: Increasing the number of wells

In this section, we will see how the algorithm performs when the number of wells, or units, increases. Six different scenarios will be considered: 10, 20, 30, 60, 80 and 100 wells. The GLPCs for the wells are shown in Fig. 8. The GLPCs themselves are not too interesting, but are included to illustrate that the wells are all represented by different concave functions. Further, the lower and upper bounds on the gas lift rates are also visible in the figure.

For each set of wells, three different starting points are considered. As opposed to before, the total available lift gas varies for each starting point, see Table 2.

The evolution of the total oil rate for the scenarios with 10, 20 and 30 wells are shown in Fig. 9. *E.g.*, the first row of subplots are the three different starting points for the case with 10 wells. A similar plot for the scenarios with 60, 80 and 100 wells are available in Fig. 10. However, due to the complexity of solving the underlying MINLP for such an high amount of wells, the optimal oil rate lines calculated by BONMIN are not included. Nonetheless, it illustrates how the proposed algorithm increases in number of required true function evaluations for an increasing amount of units before convergence is reached. The scenarios with 10, 20 and 30 wells already have illustrated that the algorithm converges to good local optima.

In both of these two plots, we can see the required number of function evaluations to reach convergence is increased with a higher number of wells. However, the increase does not appear to be exponential.

The time taken to run the algorithm for all three starting points are shown in Fig. 11. The computational time clearly grows exponentially with an increasing amount of wells.

5.2. Criticality substep design choice justification

It was mentioned in Section 2.6 that a design choice was taken based upon experience. More specifically, it was decided to reduce all the radii in case that at least one of the radii was too large compared to the corresponding gradient element. The other design option was to only reduce the corresponding radius. In the following comparison, a third option is included: Skip the criticality substep altogether. This corresponds to setting the μ_c parameter to ∞. The three options are summarized below:

- **Option 1:** Reduce all the radii if at least one element of the gradient of the Lagrangian is small compared to the corresponding radius.
- **Option 2:** Only reduce the radius for a unit if the corresponding element of the gradient of the Lagrangian is small compared to its radius.
- **Option 3:** Skip the criticality substep altogether.

A setup with 10 wells are used in this comparison. The evolution of the total oil rate is shown in Fig. 12 where each row of subplots presents one of the three options, and the columns are different starting points. It is evident that Option 2 is an inferior option for all the three starting points compared to Option 1. The performance of Option 3 is closer to the one obtained in Option 1. However, Option 1 is superior.

The reason why Option 3 is exhibiting a slower convergence than Option 1 may be explained as follows. In Option 3, all the radii will only be reduced when the if-test at line 21 of Algorithm 2 is true. *I.e.*, all the radii are reduced when all the model prediction qualities are deemed satisfied and the solution of the subproblem did not provide a better value of the true function. In contrast, Option 1 will in addition reduce all the radii if one of the relationships between the gradient elements and the radii are unsatisfactory regardless of the value of the true function.

5.3. Importance of the minimum radii parameter

A key parameter of Algorithm 2 regarding convergence of the algorithm is $\Delta_{\min}$. In this section we will illustrate two different aspects of this parameter. From one aspect, a smaller value of $\Delta_{\min}$ is beneficial, whereas the other aspects advocates a larger value. To illustrate its impact on the performance of the algorithm, the setup used in Section 4.3 with the second set of starting points (Starting point 2) will be used. Recall that the algorithm converged with a sub-optimal set of units being on/off for that case. This is acceptable as it is not a global algorithm. In the examples below, the algorithm also converges to sub-optimal sets with regards to the binary variables. For each example, the optimal gas lift rates with respect to the final set of binary variables are used as the dotted lines in the figures.

The parameters used for the following examples are found in Table 1 except for the initial radii $\Delta_0 \in \{500, 2000\}$ and the minimum radii $\Delta_{\min} \in \{0, 50, 200\}$.

In the first example, the initial start radii are set to 2000. The resulting evolution of the gas lift rates and the total oil rate for the three different minimum radii values are plotted in Fig. 13 and Fig. 14, respectively. Both figures show that a smaller value of $\Delta_{\min}$ yields a more desirable response. A higher value of the parameter results in a zigzagging response of the gas lift rates, see Fig. 13. As a result of this alternating response around the optimal actions, the resulting total oil rates is negatively impacted, see Fig. 14. A tempting conclusion from this small example is to set $\Delta_{\min} = 0$. In the next example we will illustrate why this is discouraged.

In the second example, the initial start radii are set to 500, and the same types of plots are available in Figs. 15 and 16. It becomes evident why setting Δ_0 to zero is discouraged. The algorithm converges to a point, but it is not a (local) optimum. The radii are going too fast to zero which then hinder the algorithm to move further in the solution space. When the minimum radii is increased to 50 and 200, the responses are as expected. The algorithm converges towards the (local) optimum. The larger $\Delta_{\min}$ gives a faster response, but also ends with more zigzagging around the optimum.

These two examples have illustrated the importance of the $\Delta_{\min}$ regarding safeguarding against unwanted convergence, or stopping, of the algorithm due to the Δ approaching zero.

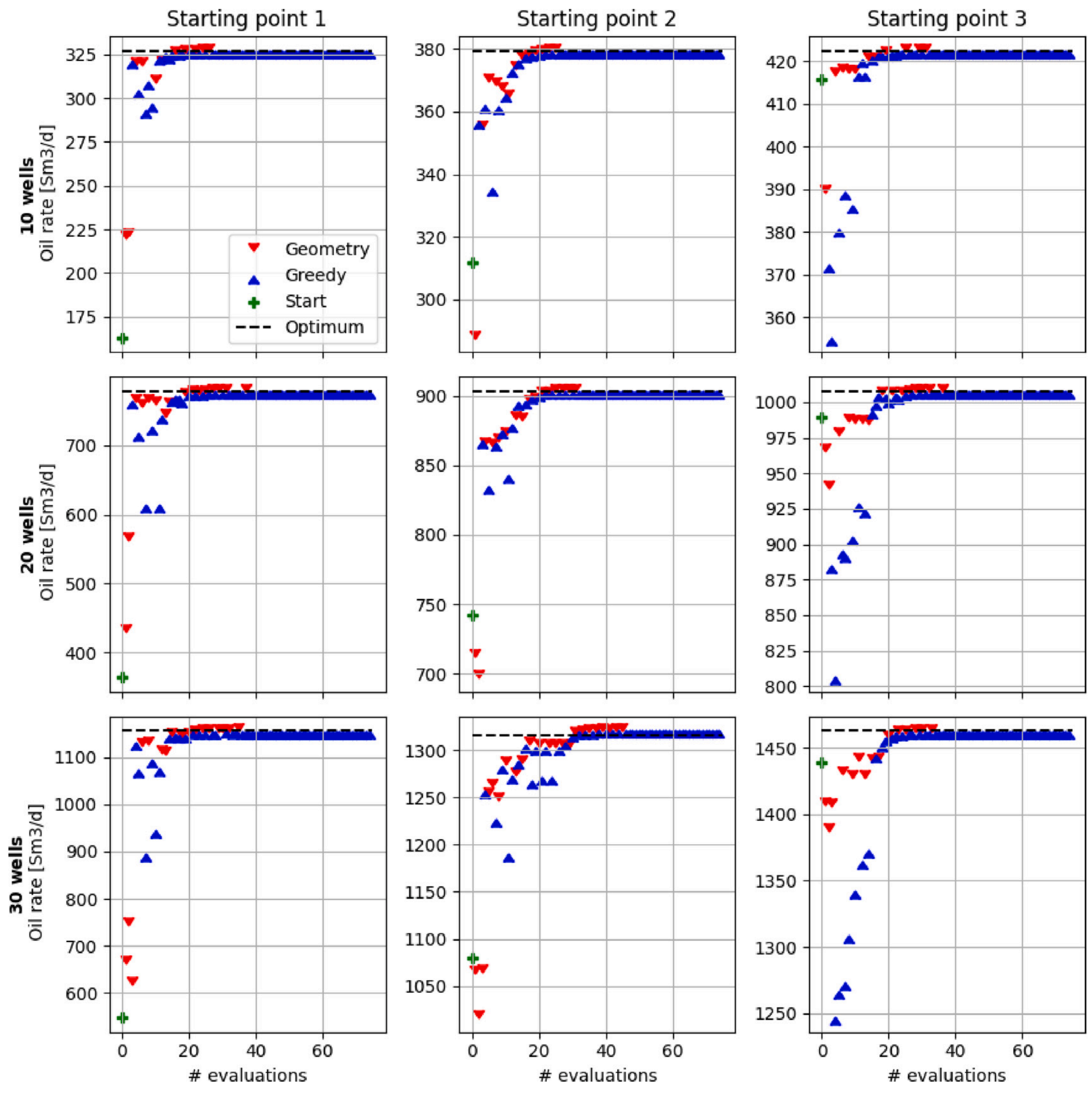

Fig. 9. The evolution of the total oil rate for the setups with 10, 20 and 30 wells. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry. The available gas is different for each starting point.

5.4. Parameters impacting the evolution of the radii

The examples in Section 5.3 made it evident that the $\boldsymbol{\Delta}_{\min}$ parameter is of high importance regarding the behavior and convergence of the algorithm. In this section, we will discuss some of the other parameters of the algorithm and how they impact the radii and its convergence. The example with starting radii of 500, and $\Delta_{\min} = 0$ in the previous section is used as a base case. The algorithm did not converge to a local optimum in the base case, and we will here show how the choices of other parameters will impact the evolution of the radii.

The two most obvious parameters that will directly impact the radii are the contraction factors: γ and ω_c. Setting these parameters closer to one will ensure that the radii are reduced slower. On the other hand, a lower value will result in a more aggressive reduction of the radii. In the first column of subplots in Fig. 17, the evolution of the oil rate and the radii are shown for the case with high contraction factors, $\gamma = \omega_c = 0.95$. As opposed to the base case, the algorithm converges to the local optimum as the radii does not reach zero before the local optimum is approached.

Another parameter is the threshold on what is considered a sufficient model prediction: η_1. The result of tuning this parameter is less obvious. If the value is almost one, then most predictions will be deemed unsatisfactory. This means that the corresponding radii will be reduced. However, because one or more predictions are bad, then neither the criticality substep nor the "reduce all" functionality at line 21 of the algorithm will happen. The evolution with $\eta_1 = 0.999$ is shown in the second column in Fig. 17. Compared to the base case, the algorithm reaches closer to the local optimum, but also here it is stopped by the radii going to zero.

Yet another parameter that could help slowing down the reduction of the radii is the μ_c. By setting this parameter high enough, the criticality substep will never happen. However, as we saw in Section 5.2, the additional reduction resulting from this step did result in better performance. Nonetheless, with a high enough value of μ_c, one or more radii will only be reduced if (i) a model predicts badly, or (ii) all models predicts well and there was no improvement in the true function. The evolution with μ_c set to its default value squared is shown in the third column in Fig. 17. In this case, the local optimum is reached. The radii are kept constant for several iterations whilst the algorithm keeps improving the value of the true function.

5.5. Randomized starting points

In this section, we will quantitatively investigate if the starting point is of importance regarding experienced convergence.

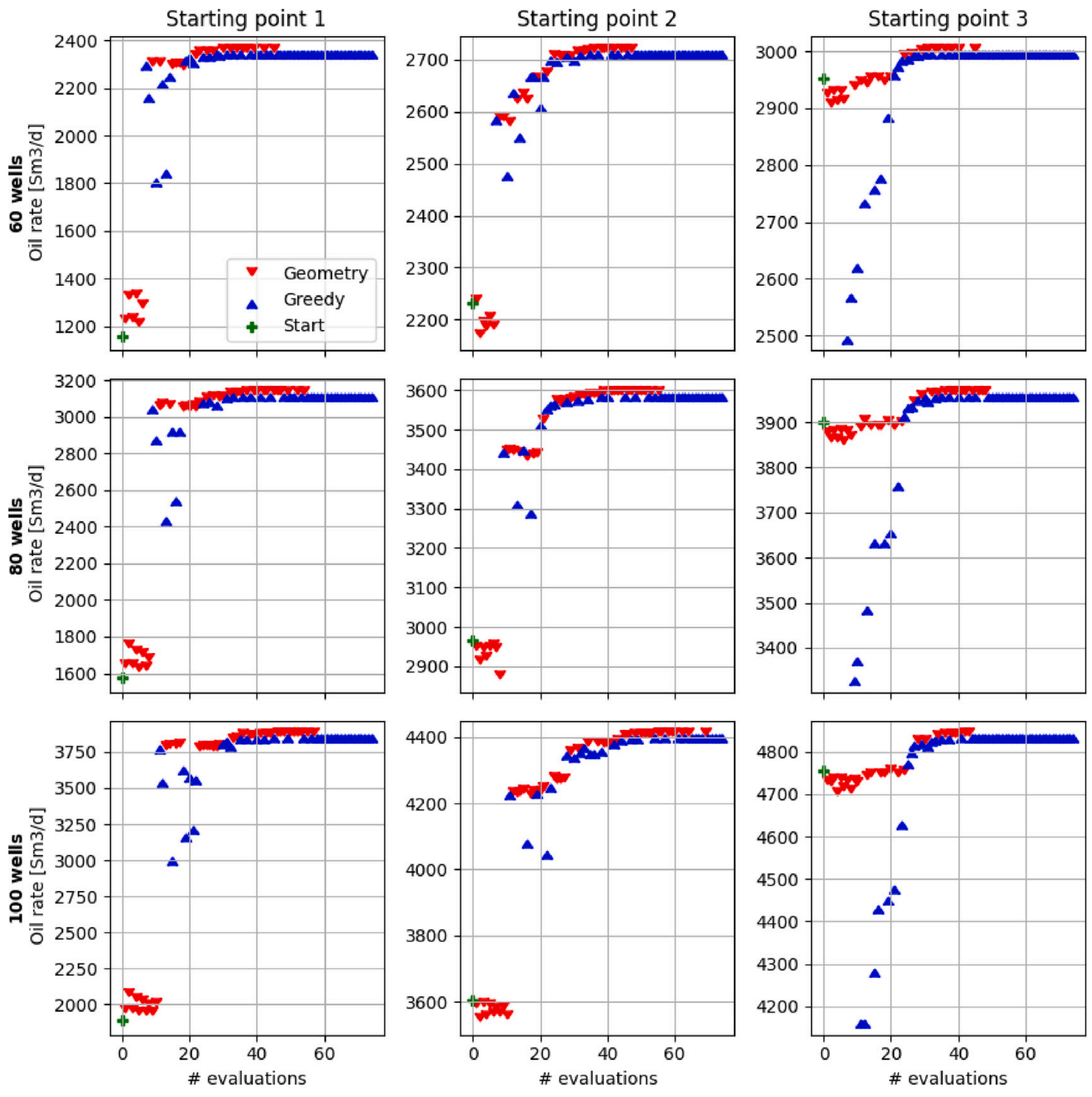

Fig. 10. The evolution of the total oil rate for the setups with 60, 80 and 100 wells. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry. The available gas is different for each starting point.

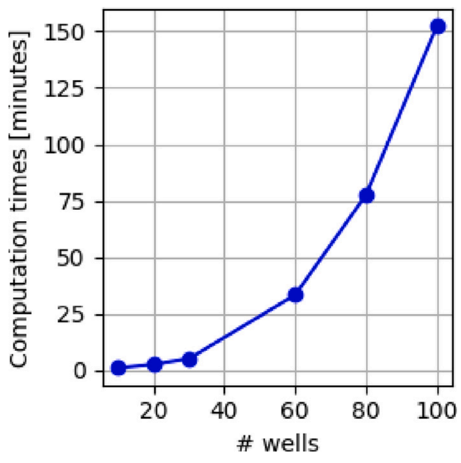

Fig. 11. The computational time to solve 75 evaluations of the algorithm for three different starting points with an increasing number of wells.

A setup with 10 wells, and a 100 randomized starting points are used. All of the starting points have the same amount of available lift gas. For all starting points, the available lift gas is given as the sum of all the minimum gas lift rates multiplied by 1.2. Each well gets its minimum gas lift rate, and the remaining 20% is uniformly randomly divided between the 10 wells. The assigned initial gas lift rates were observed to always be below the upper limits for all wells.

The results are provided in Figs. 18 and 19. For each starting point, we checked the obtained best greedy point at several checkpoints. The checkpoints were set to 10, 15 and 25 evaluations of the true function. By "best greedy point" it is meant the best point observed so far amongst the points that were found by solving the subproblem (8), *i.e.*, ignoring the geometry improvement points. Two metrics to discuss convergence are used. First, in Fig. 18, the total oil production of the best point is compared to the one found by solving the same problem with BONMIN. In this metric, a point is counted as successful if it has 99% or more of the total oil rate found by BONMIN. Second, in Fig. 19, the z variable of the best points are considered. In this metric, a point is counted as successful if it has the same z-vector as the one found by BONMIN.

As we can see in the two figures, in 99 out of the 100 different starting point, the algorithm converges to a point very close to the point

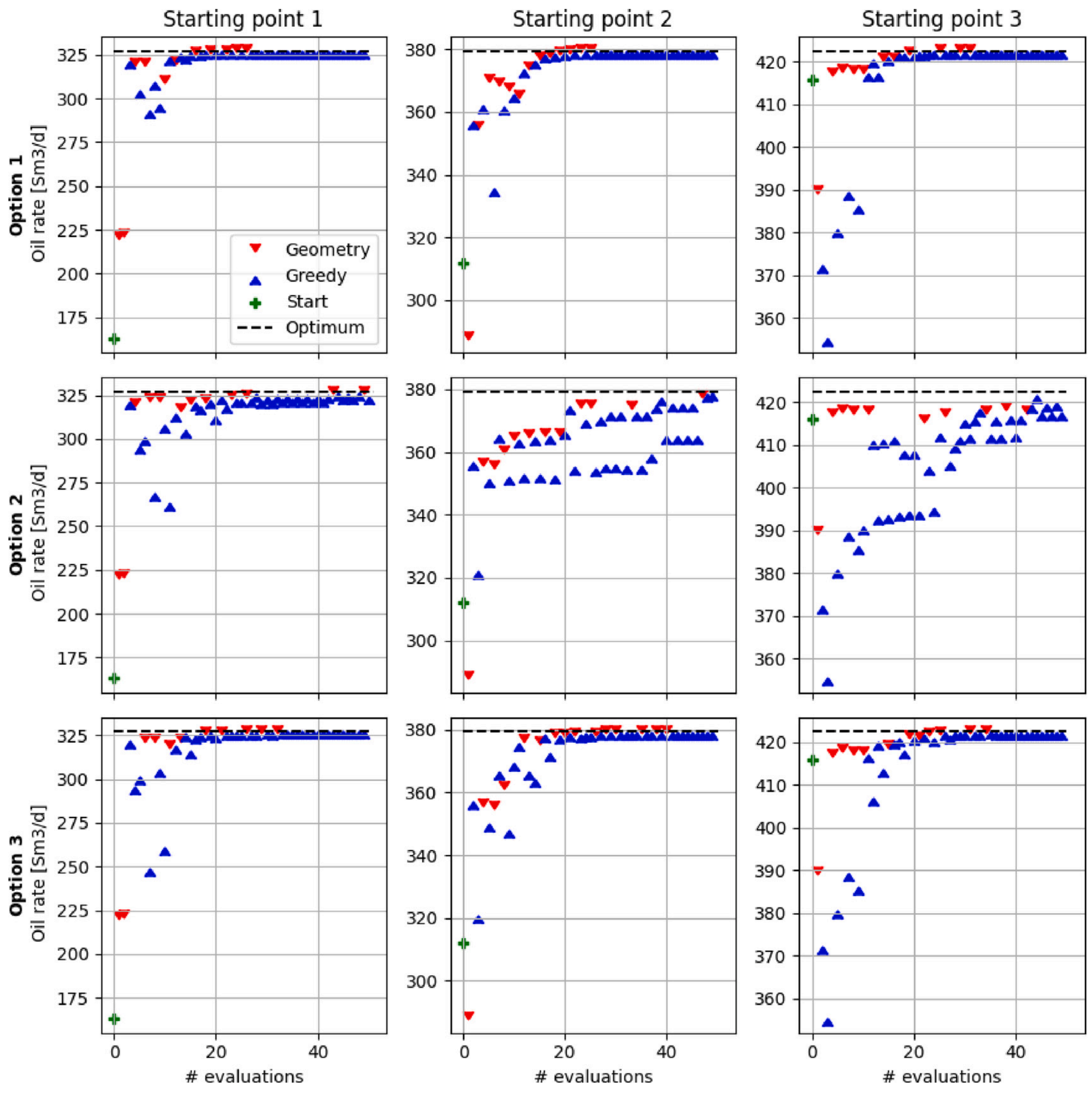

Fig. 12. The evolution of the total oil rate for the setup with 10 wells in Section 5.2. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry. The available gas is different for each starting point.

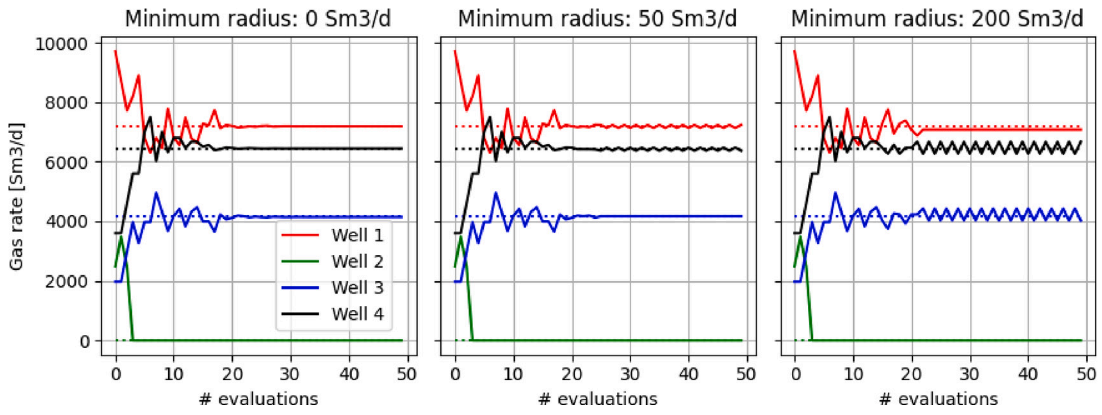

Fig. 13. The evolution of the gas lift rates for different minimum radii values and with starting radii set to $\Delta_0 = 2000$.

found by BONMIN. In only 1 case does the algorithm converge to a solution with an incorrect z-variable, see Fig. 19.

The reason why the first metric uses a comparison percentage less than 100, is that the algorithm is not expected to converge exactly

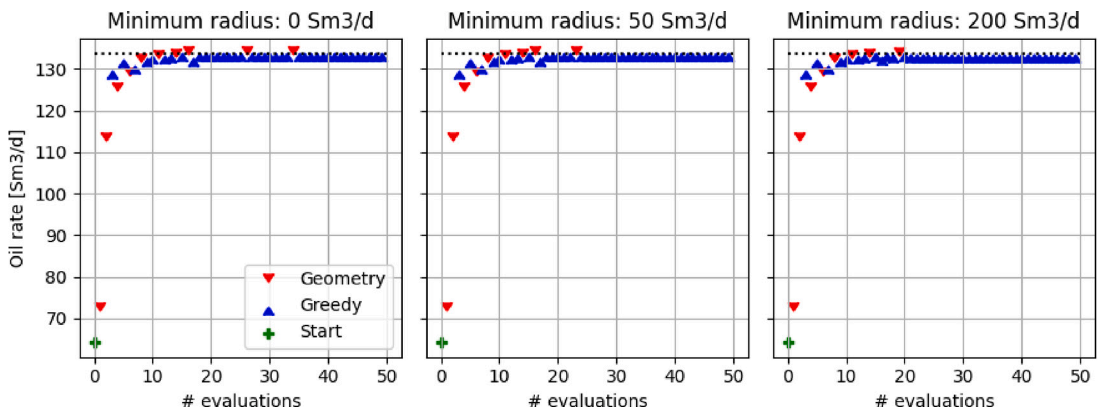

Fig. 14. The evolution of the total oil rate for different minimum radii values and with starting radii set to $\Delta_0 = 2000$. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry.

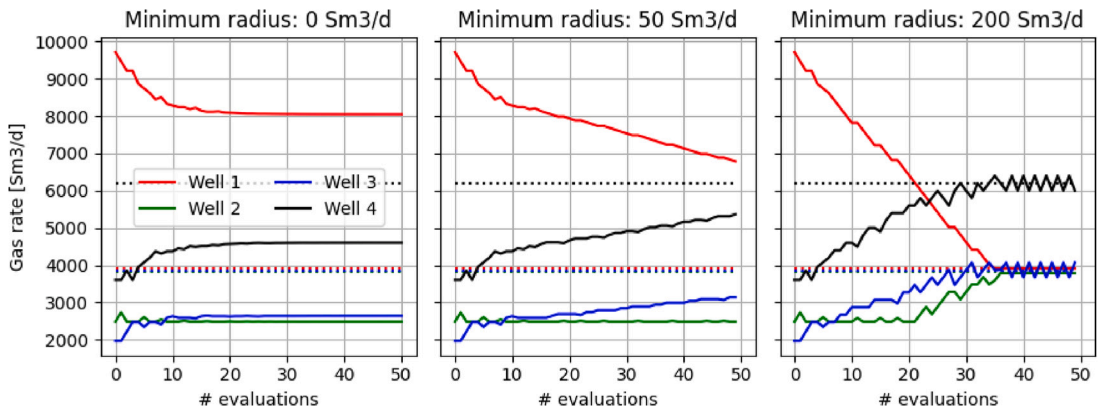

Fig. 15. The evolution of the gas lift rates for different minimum radii values and with starting radii set to $\Delta_0 = 500$.

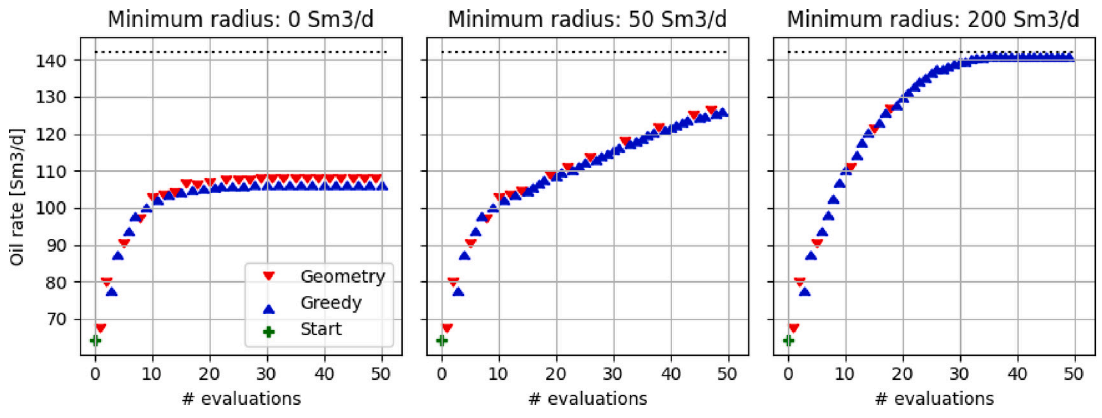

Fig. 16. The evolution of the total oil rate for different minimum radii values and with starting radii set to $\Delta_0 = 500$. The "Greedy" points are points found solving the subproblem. The "Geometry" points are points found to improve the geometry.

to the (local) optimum. This is due to the $\Delta_{\min}$ parameter, and the zigzagging behavior that follows from it, see the rightmost plot in Fig. 13.

6. Discussion

The proposed algorithm has been designed to tackle a resource allocation problem where the most challenging obstacle was that the relationship between the allocated resource and the corresponding cost of a unit was unknown. The lack of models motivated the use of a data-driven derivative-free method. In the previous section, it was illustrated by examples that the algorithm converges to local optima. In 99 out of 100 cases when considering different starting points, the final vector of binary variables were the same as the presumably globally optimal solution found by BONMIN. However, the algorithm is not designed to be a global optimizer, and converging to a local solution is expected behavior. Moreover, when the binary variable vector, **z**, was kept fixed, the algorithm converged to the local optimum in all cases.

An attractive feature of the algorithm is that it seems the number of required evaluations of the true function is moderate, and does not grow exponentially with the number of units. A hidden cost when looking at the convergence time, is the time taken to solve all the optimization formulations within the algorithm. The geometry improvement optimization formulation (32) is a Mixed Integer Nonlinear

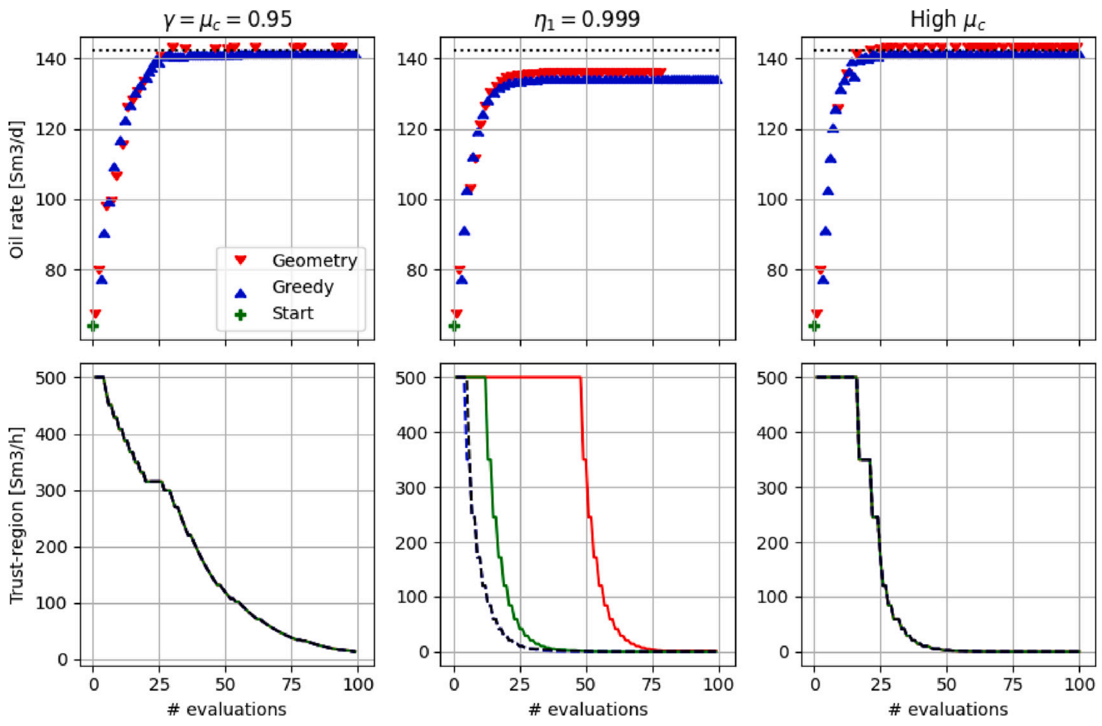

Fig. 17. The behavioral differences of the algorithm due to selected changes of the default parameters. The applied parameters are the default ones given in Table 1, except for those noted in the titles of the subplots and the starting radii which were set to 500 for all three cases.

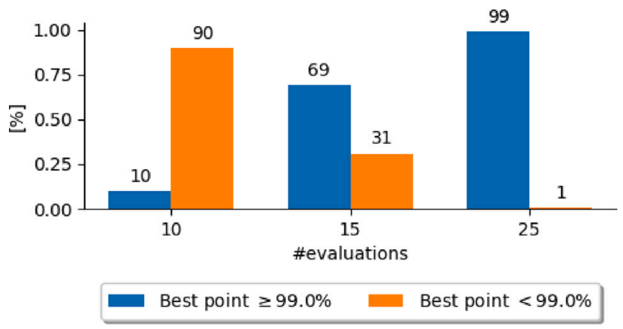

Fig. 18. The blue bars indicate the number of instances where the best greedy point found were no worse than 1% of the point found by bonmin. The x value specifies the checkpoints in evaluations. The orange bars are the opposite of the blue.

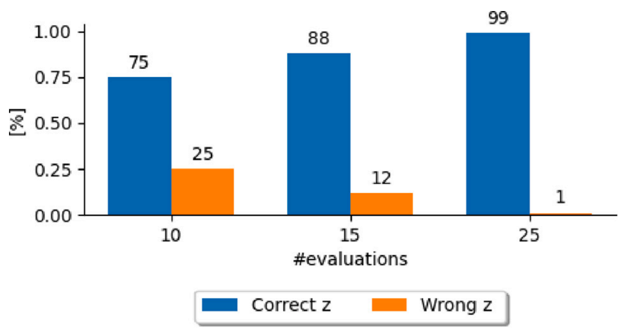

Fig. 19. The blue bars indicate the number of instances where the z value corresponding to the best greedy point found were the same as the one found by bonmin. The x value specifies the checkpoints in evaluations. The orange bars are the opposite of the blue.

Program (MINLP) which is a complicated problem structure to solve. Nonetheless, in this work the true function evaluations are considered to be the most time consuming task. In the case study, after a set of gas lift rates are imposed, it is required to wait for a steady-state of the system before measurements may be taken. The transition time from a steady-state to the next is expected to be in the range of several hours. The increase in computational time is an expected result as a MINLP solver typically exhibits such a phenomenon. Nonetheless, we only need to solve either the geometry improvement optimization problem or the subproblem once before the inputs are applied to the system.

A disadvantage of this tailored algorithm is that a proof along the lines of Conn et al. (2009) to show global convergence to first-order critical points does no longer apply as the proposed method deviates from the framework in Conn et al. (2009). The major differences being the single pass criticality substep, as explained in Section 2.6, the required minimum trust-region radii parameter $\Delta_{\min}$ to avoid the radii going to zero, and the use of several trust-region radii and models. These changes are not aligned with the framework of Conn et al. (2009), and thus, the proof of convergence cannot be applied. Nonetheless, the proposed method contains the same type of building blocks as the one in Conn et al. (2009). Furthermore, global convergence was experienced in all examples.

A final advantage of the method is that it only evaluates the true function at points that are feasible with respect to the underlying problem (1). This is the case as the constraints are imposed at all times when any new point is selected.

7. Conclusion

We proposed a derivative-free trust-region model-based algorithm to tackle a resource allocation problem with some specific characteristics that were inspired by a real world problem. The algorithm converged to local optima in all the compared cases.

In addition, we proposed a decomposition of the gas lift performance curve to exploit available sensor data. Furthermore, we illustrated how this decomposition can be combined with the suggested algorithm.

CRediT authorship contribution statement

Joakim R. Andersen: Conceptualization, Methodology, Software, Validation, Writing – original draft, Visualization. **Lars Imsland:** Conceptualization, Writing – review & editing, Supervision. **Alexey Pavlov:** Conceptualization, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research is a part of BRU21 – NTNU Research and Innovation Program on Digital and Automation Solutions for the Oil and Gas Industry (www.ntnu.edu/bru21).

Partial financial support from Wintershall Dea Norge AS is acknowledged. They had no involvement regarding content nor writing of this article.

We acknowledge Mammad Mirzayev for stimulating discussions on the industrial task that inspired this work.

References

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. CasADi – A software framework for nonlinear optimization and optimal control. Math. Program. Comput. 11 (1), 1–36. http://dx.doi.org/10.1007/s12532-018-0139-4.

Bajaj, I., Hasan, M.F., 2019. UNIPOPT: Univariate projection-based optimization without derivatives. Comput. Chem. Eng. 127, 71–87. http://dx.doi.org/10.1016/j.compchemeng.2019.05.008.

Bajaj, I., Iyer, S.S., Faruque Hasan, M., 2018. A trust region-based two phase algorithm for constrained black-box and grey-box optimization with infeasible initial point. Comput. Chem. Eng. 116, 306–321. http://dx.doi.org/10.1016/j.compchemeng.2017.12.011.

Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A., 2008. An algorithmic framework for convex mixed integer nonlinear programs. Discret. Optim. 5 (2), 186–204, In Memory of George B. Dantzig. http://dx.doi.org/10.1016/j.disopt.2006.10.011.

Conn, A.R., Scheinberg, K., Vicente, L.N., 2009. Introduction to Derivative-Free Optimization. SIAM, http://dx.doi.org/10.1137/1.9780898718768.

Giuliani, C.M., Camponogara, E., 2015a. Derivative-free methods applied to daily production optimization of gas-lifted oil fields. Comput. Chem. Eng. 75, 60–64. http://dx.doi.org/10.1016/j.compchemeng.2015.01.014.

Giuliani, C.M., Camponogara, E., 2015b. Derivative-free optimization with use of problem structure: Applications to oil production. In: 2015 IEEE Int. Conf. Autom. Sci. Eng. CASE, pp. 764–768. http://dx.doi.org/10.1109/CoASE.2015.7294173.

Giuliani, C.M., Camponogara, E., Plucenio, A., 2013. A computational analysis of nondifferentiable optimization: Applications to production maximization in gas-lifted oil fields. In: 2013 IEEE Int. Conf. Autom. Sci. Eng. CASE, pp. 286–291. http://dx.doi.org/10.1109/CoASE.2013.6653975.

Katoh, N., Shioura, A., Ibaraki, T., 2013. Resource allocation problems. In: Handbook of Combinatorial Optimization. Springer New York, New York, NY, pp. 2897–2988. http://dx.doi.org/10.1007/978-1-4419-7997-1_44.

Krishnamoorthy, D., Foss, B., Skogestad, S., 2016a. Real-time optimization under uncertainty applied to a gas lifted well network. Processes 4 (4), http://dx.doi.org/10.3390/pr4040052.

Krishnamoorthy, D., Pavlov, A., Li, Q., 2016b. Robust extremum seeking control with application to gas lifted oil wells. IFAC-PapersOnLine 49 (13), 205–210. http://dx.doi.org/10.1016/j.ifacol.2016.07.952, 12th IFAC Workshop Adapt. Learn. Control Signal Process. (ALCOSP) 2016.

Nocedal, J., Wright, S.J., 2006. Numerical Optimization. Springer New York, New York, NY, http://dx.doi.org/10.1007/978-0-387-40065-5.

Peixoto, A.J., Pereira-Dias, D., Xaud, A.F., Secchi, A.R., 2015. Modelling and extremum seeking control of gas lifted oil wells. In: 2nd IFAC Workshop Autom. Control Offshore Oil Gas Prod. (OOGP) 2015. IFAC-PapersOnLine 48 (6), 21–26. http://dx.doi.org/10.1016/j.ifacol.2015.08.004.

Powell, M.J., 1994. A direct search optimization method that models the objective and constraint functions by linear interpolation. In: Advances in Optimization and Numerical Analysis. Springer, pp. 51–67.

Powell, M.J., 2009. The BOBYQA algorithm for bound constrained optimization without derivatives. Cambridge NA Report NA2009/06, University of Cambridge, Cambridge 26.

Rashid, K., 2010. Optimal allocation procedure for gas-lift optimization. Ind. Eng. Chem. Res. 49 (5), 2286–2294. http://dx.doi.org/10.1021/ie900867r.

Rashid, K., Bailey, W.J., Couet, B., 2012. A survey of methods for gas-lift optimization. Model. Simul. Eng. 2012, 16. http://dx.doi.org/10.1155/2012/516807.

Wächter, A., Biegler, L.T., 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. 106 (1), 25–57. http://dx.doi.org/10.1007/s10107-004-0559-y.