# Surrogate Assisted Monte Carlo Tree Search in Combinatorial Optimization

**Saeid Amiri**[1], **Parisa Zehtabi**[2], **Danial Dervovic**[3], **Michael Cashmore**[3]
[1]JP Morgan AI Research. New York City, NY, USA
[2]JP Morgan AI Research. London, UK
[3]JP Morgan AI Research. Edinburgh, UK
saeid.amiri@jpmchase.com, parisa.zehtabi@jpmorgan.com, danial.dervovic@jpmchase.com, michael.cashmore@jpmorgan.com

## Abstract

Industries frequently adjust their facilities network by opening new branches in promising areas and closing branches in areas where they expect low profits. In this paper, we examine a particular class of facility location problems. Our objective is to minimize the loss of sales resulting from the removal of several retail stores. However, estimating sales accurately is expensive and time-consuming. To overcome this challenge, we leverage Monte Carlo Tree Search (MCTS) assisted by a surrogate model that computes evaluations faster. Results suggest that MCTS supported by a fast surrogate function can generate solutions faster while maintaining a consistent solution compared to MCTS that does not benefit from the surrogate function.

## Introduction

As populations shift, market trends change, and customer demands evolve, many service industries and retail stores are faced with the decision of adding, removing, relocating, or consolidating their facility locations. An example is a 2018 survey that showed the market trends resulting in some liquor stores becoming obsolete [1]. In this paper, we focus on a particular class of facility location problem that involves closing a fixed number of retail stores in which computing the features of the evaluation function is expensive. This problem is a Combinatorial Optimization (CO). COs are often NP-hard and computationally intractable due to the large state-spaces. Consequently, solving CO problems often requires designing heuristics or approximation algorithms (Williamson and Shmoys 2011). Furthermore, real-world optimization problems are often complex, nonlinear, and may have multiple objectives and constraints that can be computationally expensive to evaluate. The solutions to CO often involves the design of heuristics or approximation algorithms.

Monte Carlo Tree Search (MCTS) (Kocsis and Szepesvári 2006; Coulom 2006) is a popular technique for solving search problems in large spaces, particularly in the domain of games. It involves building a search tree of possible actions and their corresponding outcomes, and using evaluations (simulations) to estimate the value of each action. MCTS has been applied to a wide range of problems such as games (Silver et al. 2017; Rubin and Watson 2011), robotics (Kim et al. 2020), finance (Vittori, Likmeta, and Restelli 2021) and music (Liebman et al. 2017). Recent works have used MCTS in CO. One work used Graph Neural Networks and Reinforcement Learning in order to compute heuristics for the MCTS-based action selection in scheduling and vehicle routing problems (Oren et al. 2021). MCTS has also been applied to capacity expansion in a residency matching problem to find an optimal policy for matching medical doctors to hospital vacancies (Abe, Komiyama, and Iwasaki 2022).

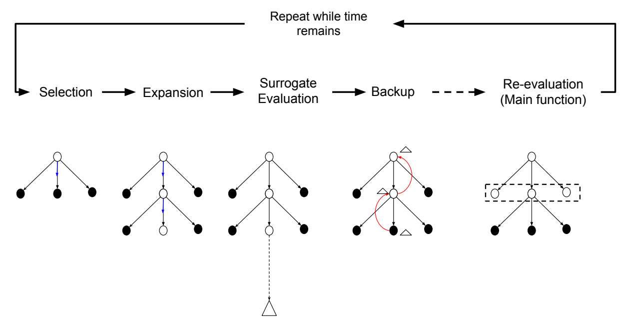


Figure 1: Surrogate assisted Monte Carlo Tree Search (SMCTS) where an occasional reevaluation step refines the node values.

Inspired by these successes, we leverage MCTS in the facility location problem. We propose Surrogate-assisted MCTS (SMCTS) to solve a combinatorial search problem where we use a fast surrogate evaluation function in concert with the slow default evaluation function. The main evaluation function is a regression model that evaluates the current network profitability but is computationally expensive due to the varying network-dependent features. The surrogate is fast to compute but is less accurate. The choice of an efficient surrogate function is its own research problem, and we simply assume that a surrogate function is available. In this



[1]https://www.forbes.com/sites/taranurin/2019/11/22/independent-liquor-stores-will-become-obsolete-believe-nearly-one-out-of-two-owners/?sh=5eb971f57b9b

paper, we focus on how to use the surrogate function jointly with the main evaluation function, aiming at faster solution computation. Figure 1 depicts the SMCTS steps where the selection, expansion, evaluation (by surrogate function), and backup is complemented by an occasional re-evaluation step that takes place in order to refine possible inaccurate surrogate model evaluations.

We apply this approach to the problem of store closure in a network of liquor stores with the goal of minimizing the overall sales loss. Our empirical results show that by using MCTS with a surrogate function, the overall computation time is reduced.

## Related Work

Classes of facility location problems are among the fundamental problems in Operations Research. Traditionally, they have been framed using Operation Research techniques such as set covering (Namazian and Roghanian 2021; Murray 2016; Miliotis, Dimopoulou, and Giannikos 2002), maximal covering (Church and ReVelle 1974; Berman and Krass 2002), or p-median problems where the goal is minimizing the travel distance from customers (Kariv and Hakimi 1979). Facilities could be static such as branches and warehouses (Zaikin, Ignatiev, and Marques-Silva 2020) or dynamic such as charging stations (Andrenacci, Ragona, and Valenti 2016; Drezner and Wesolowsky 1991; Wesolowsky 1973).

Most works formulate the problem either as integer programming or clustering methods. In the former, various heuristic techniques such as tabu search, Lagrangian relaxation (Santos 2009), greedy interchange (Kuehn and Hamburger 1963), branch and bound (Järvinen, Rajala, and Sinervo 1972; Dupont 2008), primal and dual (Captivo 1991), binomic approach (Maniezzo, Mingozzi, and Baldacci 1998) and gamma heuristics (Rosing, Revelle, and Schilling 1999) have been used. In the latter, some works leveraged unsupervised clustering methods to find the demand centroids for charging stations and assign customers based on distance to the closest centroid (Andrenacci, Ragona, and Valenti 2016; Ip, Fong, and Liu 2010). In some papers, Analytic Hierarchy Process have been proposed for banking branch location based on customer demographic and economic features (Gorener, Dinçer, and Hacioglu 2013; Sharmin and Solaiman 2019).

Most works have used the travel distance and customer demand as the only features in their objective and constraints formulation. Zaikin, Ignatiev, and Marques-Silva (2020) set the customer dissatisfaction minimization as the goal in the branch closure problem using Max-SAT methods.

Almost all works have considered the solution to the facility location problem as a one-shot static solution where the location of multiple facilities are selected all at once. In practice, firms and industries would decide to alter the exisiting network of facilities. One work considered removing or addition of existing facilities using integer programming and approximation techniques (Wang et al. 2003).

In comparison, this paper focuses on a data-driven evaluation function that estimates the overall sales profit based on numerous features such as the number and the distance of nearby facilities. In addition, we leverage surrogate evaluations as a fast evaluation for the facility location problems.

**Surrogate-assisted optimization** Leveraging surrogates for optimization has already been explored by researchers in other contexts (Namazi et al. 2020; Gu et al. 2021; Liu, Zhang, and Gielen 2013). For instance, (Liu, Zhang, and Gielen 2013) proposed a Gaussian Process-assisted evolutionary algorithm to solve computationally expensive problems. A surrogate model has been used to prune the solution search space in the Travelling Thief problem (Namazi et al. 2020).

To the best of our knowledge, this is the first work of leveraging surrogates in MCTS in CO problems.

## Problem Statement

This problem is a class of facility location problem in which a fixed number of retail stores are going to be closed. There is a city network of $N \in \mathbb{N}$ stores. We seek to remove $M$ stores ($M < N$), that result in minimum forgone sales of the network. Our decision variable is the vector $X$ ($|X| = N$) such that:

$$X_j = \begin{cases} 1 & \text{if store } j \text{ remains open} \\ 0 & \text{If store } j \text{ remains closed} \end{cases}$$

and the objective is expressed as:

$$\text{Minimize} \sum_{j=1}^{N} F_m(\mathbb{1}_N, j) - \sum_{j=1}^{N} F_m(X, j)$$

$$\text{Subject to } ||X||_2 = M$$

where $\mathbb{1}_N$ is a vector of 1s and size $N$. The objective as mentioned above is to minimize the total loss of sales as a result of store closures and the constraint states that exactly $M$ stores will be closed. $F_m$ is an evaluation function that estimates the sales of store $j$. It is important to note that the sales estimated per store depend not only upon the features of that store, but also on other stores, including whether or not they are closed.

Next, we describe how we find the solutions to this optimization problem.

## Framework

In this section, we explain the surrogate assisted MCTS framework.

**Node representation:** In our search tree, a node is identified by the set of candidate stores for removal according to the path from the root. The root node represents no store removal and the tree depth is $M$ where $M$ is the total number of stores to be removed from the network. In other words, the nodes at depth $M$ are terminal nodes. Each node keeps duplicate attributes $V_s'$ and $N_s'$ for its value and the number of visits in case it goes through the re-evaluation step. SMCTS has five components:

- **Selection:** Starting at the root node, the tree policy, in our case UCB1 (Kocsis and Szepesvári 2006), is used to select the next node based on its value, as described below:

$$\text{argmax}\left\{v_s + C\sqrt{N_p/N_s}\right\}$$

  where $v_s$ is the value of the node $s$, $N_s$ the number of times node $s$ is visited, $N_p$ is the number of times the parent node has been visited. The choice of $C$ affect the ratio of exploration versus exploitation in the search.[2]
- **Evaluate:** A node can be evaluated using functions $F_m$ and $F_s$. $F_m$ is the main evaluation function that is costly to compute. $F_s$ is an approximate surrogate function that is faster to compute but less accurate compared to $F_m$.
- **Backup:** The return generated by the main or surrogate evaluation function is backed up to update the values.
- **Expand:** A node is expanded to its children by removing any of the remaining stores from the network. The number of children expanded is equal to the remaining number of stores in the network. Removing a store can be denoted as taking action $a_i \in A$ meaning the removal of the $i$th store.
- **Re-evaluation:** A node's children are re-evaluated if their values are within the estimation error of the neighboring nodes in the same tier.

Next, we briefly explain the SMCTS algorithm.

**Algorithm:** Algorithm 1 requires a surrogate function $F_s$ with the error bound $\sigma_s$, (in our case, $\sigma_s$ is the difference between the Root Mean Squared Error (RMSE) of the $F_s$ and $F_m$) and a main evaluation function is $F_m$. Node $s$ is initialized with the root node $s_0$. The selection is done using the UCB1 algorithm where it suggests the best action $a$ denoting the next best store for removal (Lines 4-5). Once the next node is selected, it expands into new children. (Lines 6 -7). The value of the node is estimated by the surrogate function $F_s$ and backed up to the parent nodes recursively (Lines 9-10). The novelty of SMCTS is in the **re-evaluate** step where an occasional refinement of node values is done in order to reduce value errors. The re-evaluation step is presented in Algorithm 2. This algorithm is called when all the children of node $s$ are visited an equal number of times. In that case, Algorithm 2 sorts the values of all children in the subtree (sharing same parent node). We name the values of two adjacent sorted nodes $V_{s_i}$ and $V_{s_{i+1}}$. These values may not be accurate as they have been evaluated using $F_s$, therefore if $V_{s_{i+1}} - \sigma_s$ is less than $V_{s_i} + \sigma_s$, then these node values need to be updated with $F_m$. The number of times that re-evaluate is called would depend on $\sigma_s$ and the distribution of node values.

## Experiments

This section details the conducted experimental evaluation, analysing the performance of SMCTS in different problem settings. Our evaluation aimed at studying the following hypothesis:

[2] We use $v'_s$, $N'_p$, $N'_s$ instead, after the node is reevaluated.

Algorithm 1: Surrogate-assisted MCTS

**Input**: Surrogate function $F_s$, evaluation function $F_m$, action set $A$, root node $s_0$, error bound $\sigma_s$

```
1: while Computational budget do
2:   s ← s_0
3:   while s.terminal is False do
4:     a ← Select(s, A)
5:     s ← s.children[a]
6:     if s.leaf is True then
7:       Expand(s)
8:     end if
9:     v ← Evaluate(s, F_s)
10:    Backup(s, v)
11:    if s.leaf is False and s.children equally visited then
12:      Re − evaluate(s, F_m, σ_s)
13:    end if
14:  end while
15: end while
16: return Node with the highest value
```

Algorithm 2: Re-evaluate nodes

**Input**: Node $s$, $F_m$, $\sigma_s$ error bound

```
Sort children of s based on value
for i in [0, s.children.length − 1] do
  v_{s_i} ← s.children[i].value
  v_{s_{i+1}} ← s.children[i + 1].value
  if v_{s_{i+1}} − σ_s < v_{s_i} + σ_s then
    v'_{s_{i+1}} ← Evaluate(s.children[i + 1], F_m)
    v'_{s_i} ← Evaluate(s.children[i], F_m)
    Backup(s_i, v'_{s_i})
    Backup(s_{i+1}, v'_{s_{i+1}})
  end if
end for
```

1. In scenarios with higher scale (large number of total stores or large number of removals), SMCTS tends to leverage surrogate function more than the evaluation function.
2. The number of surrogate evaluations depends on the surrogate quality. The higher error it has, the more re-evaluation steps are needed.
3. With an efficient choice of a surrogate function, SMCTS maintains a solution consistent with unassisted MCTS.

Next we explain the dataset and the evaluation functions.

**Dataset:** We use the Iowa Liquor Dataset [3] that contains the daily purchase information of various liquors in each store in the state of Iowa. The dataset has the information of the stores such as the store name, address, coordinates, zip code with 978 unique values, and the city name with 476

[3] https://data.iowa.gov/Sales-Distribution/Iowa-Liquor-Sales/m3tr-qhgy

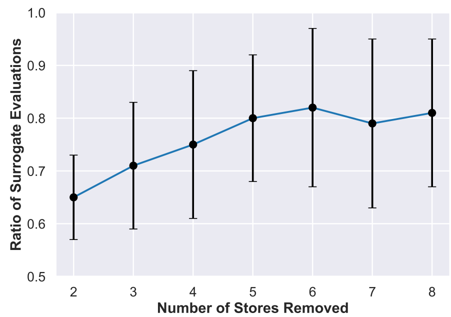


Figure 2: SMCTS where an occasional reevaluation step refines the node values. The horizontal axis represents the number of stores that need to be removed.

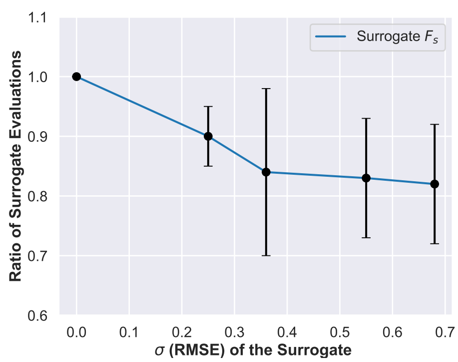


Figure 3: SMCTS with various surrogate errors. The vertical axis is the ratio of surrogate function $F_s$ evaluations to the total evaluations. The horizontal axis represent surrogate functions with increasing normalized RMSEs.

cities and the type and amount of the liquor they have sold at each day. We preprocess the data by calculating the total sales of all liquors at each store in a year and by defining some new features such as the number of stores in $0.5$ miles proximity. These additional features are network dependent and require to be recomputed for every node in the tree, resulting in $F_m$ to be costly.

**Evaluation Functions:** Our main evaluation function $F_m$ is an XGBoost regression model that estimates the sales amount for a store. Given a store removal, some features in the dataset need to be recalculated, resulting in a costly evaluation. To create a surrogate function $F_s$, we use a subset of the features of the dataset and train another XGBoost regression model. $F_s$ is less accurate on sales estimation compared to $F_m$. In our case, $F_s$ has a normalized RMSE of $0.27$ and $F_m$ has the RMSE of $0.16$ (both on the test set). We use SMCTS for liquor store removals for a given county with varying counts of store to remove. Figure 2 shows the average ratio of the number of times $F_s$ has been called versus $F_m$ for ten counties (randomly sampled) where the number of stores in those counties ranged from $17$ to $64$, representing $70$ store removal problems. We observe that with the increase in the number of removed branches, the relative number of times the surrogate function is called increases, facilitating reduction of the overall evaluation burden. Figure 3 shows the ratio of surrogate evaluation to the total evaluation using various surrogates with various error bound. With the increase in the error of the surrogate, we observe an increase in the re-evaluation step. Such increase is valuable as long as SMCTS is consistent with MCTS store selection. Figure 4 presents the consistency comparison of the two approaches for various store removals. We use the Sørensen–Dice coefficient to measure the similarity of the results of the two methods. The values are the average of ten counties, randomly sampled from the dataset. We observe that in most cases, SMCTS output is consistent with MCTS. There is a bit of inconsistency for 3 and 4 branch removals, such inconsistencies are due to the weaker estimations of $F_S$ in outlier counties.

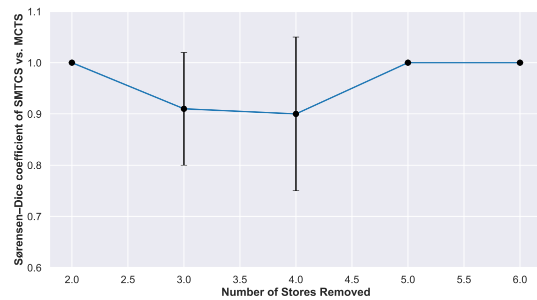


Figure 4: Evaluation of the consistency of stores selected by SMCTS vs. MCTS. The vertical axis shows the number of the selected stores by SMCTS being different from MCTS. The results are the average of $10$ counties that are randomly selected.

## Conclusion & Future Work

In this work, we proposed MCTS search with surrogate functions for combinatorial optimization. We demonstrated that by using less accurate but faster surrogate function, we can solve optimization problems more efficiently. We applied our approach to a store closure problem in which the goal is to minimize the total sales loss of a retail store.

In this paper, we assumed the surrogate function is provided while this is not the case in practice. For future work, we propose to investigate ways to implement and design the surrogate function and the criteria for it to improve the SMCTS. In addition, we will explore the applicability of SMCTS with other datasets and domains with stochasticity in the action space.

**Disclaimer.** This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates (“JP Morgan”), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

## References


Abe, K.; Komiyama, J.; and Iwasaki, A. 2022. Anytime capacity expansion in medical residency match by monte carlo tree search. *arXiv preprint arXiv:2202.06570*.

Andrenacci, N.; Ragona, R.; and Valenti, G. 2016. A demand-side approach to the optimal deployment of electric vehicle charging stations in metropolitan areas. *Applied Energy*, 182: 39–46.

Berman, O.; and Krass, D. 2002. The generalized maximal covering location problem. *Computers & Operations Research*, 29(6): 563–581.

Captivo, M. E. 1991. Fast primal and dual heuristics for the p-median location problem. *European Journal of Operational Research*, 52(1): 65–74.

Church, R.; and ReVelle, C. 1974. The maximal covering location problem. In *Papers of the regional science association*, volume 32, 101–118. Springer-Verlag Berlin/Heidelberg.

Coulom, R. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, 72–83. Springer.

Drezner, Z.; and Wesolowsky, G. 1991. Facility location when demand is time dependent. *Naval Research Logistics (NRL)*, 38(5): 763–777.

Dupont, L. 2008. Branch and bound algorithm for a facility location problem with concave site dependent costs. *International journal of production economics*, 112(1): 245–254.

Gorener, A.; Dinçer, H.; and Hacioglu, U. 2013. Application of multi-objective optimization on the basis of ratio analysis (MOORA) method for bank branch location selection. *International Journal of Finance & Banking Studies (2147-4486)*, 2(2): 41–52.

Gu, Q.; Wang, Q.; Li, X.; and Li, X. 2021. A surrogate-assisted multi-objective particle swarm optimization of expensive constrained combinatorial optimization problems. *Knowledge-Based Systems*, 223: 107049.

Ip, A.; Fong, S.; and Liu, E. 2010. Optimization for allocating BEV recharging stations in urban areas by using hierarchical clustering. In *2010 6th International conference on advanced information management and service (IMS)*, 460–465. IEEE.

Järvinen, P.; Rajala, J.; and Sinervo, H. 1972. A branch-and-bound algorithm for seeking the p-median. *Operations Research*, 20(1).

Kariv, O.; and Hakimi, S. L. 1979. An algorithmic approach to network location problems. I: The p-centers. *SIAM journal on applied mathematics*, 37(3): 513–538.

Kim, B.; Lee, K.; Lim, S.; Kaelbling, L.; and Lozano-Pérez, T. 2020. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 9916–9924.

Kocsis, L.; and Szepesvári, C. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer.

Kuehn, A. A.; and Hamburger, M. J. 1963. A heuristic program for locating warehouses. *Management science*, 9(4): 643–666.

Liebman, E.; Khandelwal, P.; Saar-Tsechansky, M.; and Stone, P. 2017. Designing better playlists with monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 4715–4720.

Liu, B.; Zhang, Q.; and Gielen, G. G. 2013. A Gaussian process surrogate model assisted evolutionary algorithm for medium scale expensive optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(2): 180–192.

Maniezzo, V.; Mingozzi, A.; and Baldacci, R. 1998. A bionomic approach to the capacitated p-median problem. *Journal of Heuristics*, 4(3): 263.

Miliotis, P.; Dimopoulou, M.; and Giannikos, I. 2002. A hierarchical location model for locating bank branches in a competitive environment. *International transactions in operational research*, 9(5): 549–565.

Murray, A. T. 2016. Maximal coverage location problem: impacts, significance, and evolution. *International Regional Science Review*, 39(1): 5–27.

Namazi, M.; Sanderson, C.; Newton, M.; and Sattar, A. 2020. Surrogate assisted optimisation for travelling thief problems. In *Proceedings of the International Symposium on Combinatorial Search*, volume 11, 111–115.

Namazian, Z.; and Roghanian, E. 2021. A decision problem for bank branch site selection: a GIS mapping perspective with maximal covering location problem: a case study of Isfahan, Iran. *International Journal of Services and Operations Management*, 39(3): 337–361.

Oren, J.; Ross, C.; Lefarov, M.; Richter, F.; Taitler, A.; Feldman, Z.; Di Castro, D.; and Daniel, C. 2021. SOLO: search online, learn offline for combinatorial optimization problems. In *Proceedings of the International Symposium on Combinatorial Search*, volume 12, 97–105.

Rosing, K. E.; Revelle, C. S.; and Schilling, D. A. 1999. A gamma heuristic for the p-median problem. *European Journal of Operational Research*, 117(3): 522–532.

Rubin, J.; and Watson, I. 2011. Computer poker: A review. *Artificial intelligence*, 175(5-6): 958–987.

Santos, A. C. 2009. *Solving large p-median problems using a Lagrangean heuristic*. Ph.D. thesis, Université Blaise Pascal (Clermont Ferrand 2).

Sharmin, S.; and Solaiman, K. 2019. BigBank: A GIS integrated AHP-TOPSIS based expansion model for banks. In *2019 SITIS*, 311–318. IEEE.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Vittori, E.; Likmeta, A.; and Restelli, M. 2021. Monte carlo tree search for trading and hedging. In *Proceedings of the Second ACM International Conference on AI in Finance*, 1–9.

Wang, Q.; Batta, R.; Bhadury, J.; and Rump, C. M. 2003. Budget constrained location problem with opening and closing of facilities. *Computers & Operations Research*, 30(13): 2047–2069.

Wesolowsky, G. O. 1973. Dynamic facility location. *Management Science*, 19(11): 1241–1248.

Williamson, D. P.; and Shmoys, D. B. 2011. *The design of approximation algorithms*. Cambridge university press.

Zaikin, O.; Ignatiev, A.; and Marques-Silva, J. 2020. Branch location problems with maximum satisfiability. In *ECAI 2020*, 379–386. IOS Press.