Beobachtungen:

Base Case Small:

* Dynamic deutlich besser als One
* One schlechtester
* Dynamic und Relaxed Dynamic am besten
* Andere Benchmarks dazwischen
* Dynamic 93.67% vs. Dyn-Relaxed 94.4% am besten
* One 69.4% vs One Relaxed 72.3% schwach
* Greedy 80.3% obwohl von Anfang an alles annehmen, aber am schnellsten
* Interval 89%, aber deutlich höhere Laufzeit
* WTP 86.2% schlechter bei unwesentlich geringerer Laufzeit
* Amazon 85.3% schlechter bei gleicher Laufzeit
* Runtime integral nur unwesentlich höher als fractional
* Dynamic länger als ex post, One Time darunter, nahe 0
* Relaxed genauso lange wie nicht-Relaxed
* Avg Simulation Time 0.0017 Sek.

VaryingB:

* Dynamic konvergiert langsam gen 1
* One konvergiert deutlich schneller gen 1; mit Knick bei b = 100
* Bei b = 1000 (\*3 = 3000 Ressourceneinheiten) und n=1000 Dynamic (98.9%) nur unwesentlich besser als One-Time (97.1%)
* Bei b = 50 (\*3 = 150 Ressourceneinheiten) und n=1000 Dynamic (89.5%) deutlich besser als One (67.0%)
* Greedy recht volatil, nähert sich eins an, bei überschüssigen Ressourcen ähnlich wie Dynamic und One-Time (da alle Kunden bedient werden und keine bewusste Auswahl nötig wie bei knappen Ressourcen)
* Interval Learner bleibt nahezu konstant (Wieso?? Gerade bei überschüssigen Ressourcen)
* One Relaxed liegt anfangs bei knappen Ressourcen über One, am Ende nahezu deckungsgleich bei überschüssigen Ressourcen
* Dynamic Relaxed analog
* WTP steigt, nähert sich „nur“ Interval Learner an
* Amazon einer der schlechtesten bei knappen Ressourcen (72.7%), bei knappen Ressourcen minimal bester Benchmark (99.0%). Bei knappen Ressourcen und Schattenpreise != 0 lohnt sich Ab-/Zuschlag folglich nicht, bei überschüssigen Ressourcen sind Schattenpreise ca. 0, daher kaum Unterschied durch Zu-/Abschläge
* Laufzeit der Ex-Post Optimierer nimmt mit zunehmender Ressourcenkapazität ab
* Dynamic und Relaxed nimmt analog auch ab, da ja nur mehrmalige Ausübung eines Ex Post Optimierers
* One-Time und Relaxed analog, nur generell viel kürzere absolute Laufzeit
* Greedy am schnellsten
* Interval nimmt absolut am meisten ab
* Generell: Ratios der Laufzeiten bleiben ungefähr konstant
* Änderung der rechten Seite ändert bei allen Benchmarks (außer Greedy) **nur** die Laufzeiten der Gurobi Optimierung, gleicher Abnahme-Faktor bei allen Benchmarks

VaryingEps:

* Dynamic nimmt relativ linear ab mit steigendem Eps (von 94.5% bei Eps = 0.001 auf 47.3% bei Eps = 0.25)
* One-Time fällt schnell, steigt schnell und fällt dann langsam ab; bei Eps > 0.075 besser als Dynamic (von 76.1% bei Eps = 0.001 auf 58.2% bei Eps = 0.25)
* Greedy, Interval, WTP unabhängig von Eps (logisch)
* One-Relaxed und Dyn-Relaxed werden mit zunehmendem Eps besser, da right-hand side ohne den modifying factor deutlich weniger restriktiv
* Amazon stets zwischen One-Time und Dynamic
* Dynamic lohnt sich insbesondere bei kleinen Eps (<5%, eher <2%)
* Eps-unabhängigen Benchmarks besser bei großen Eps
* Runtime Dynamic stufenförmig (z.B. Eps = 0.125 🡪 3 Aktualisierungen; Eps = 0.15 🡪 Ebenfalls 3 Aktualisierungen, aber mit größeren Problemen, daher mehr Zeit; Eps = 0.2 🡪 3 Aktualisierungen und noch größere Probleme; Eps = 0.25 🡪 Nur 2 Aktualisierungen, daher Abfall der Runtime
* One-Time dagegen linear wachsend, da graduell größere Probleme
* Runtime von Ex-Post, Greedy, Interval, WTP unabhängig von Eps
* One-Relaxed wie One, Dyn-Relaxed und Amazon wie Dyn
* Average Simulation und Permutation ebenfalls stufenförmig, da Summe aus konstanten und stufenförmigen Graphen => “steilere” Stufen

VaryingM:

* Dynamic und One-Time relativ robust ggü. Variationen in m
* Anders als bei b steigt mit der Anzahl Ressourcen auch die Anzahl Nachfragen, folglich Dynamic und One annähernd konstant
* Greedy nimmt ab mit zunehmendem m; potentiell da Kunden mit zunehmender Anzahl Ressourcen auch zunehmend heterogen werden
* Relaxed, Interval und WTP auch mit geringer Variation, da Anzahl Ressourcen und Nachfrage Hand in Hand steigen
* Amazon Learner konsistent schlechter als Dynamic, obwohl bei mehr Ressourcen mehr „Potential“ für den Effekt der Ab-/Zuschläge auf Schattenpreise; Eps scheint relevantere Einflussgröße zu sein
* Laufzeiten der Benchmarks sinken (!) mit steigendem n; wiederum mit ähnlichen Faktoren, daher vermutlich auf Gurobi zurückzuführen
* Avg Simulation Time und Permutation Time nehmen analog zu Benchmarks mit steigendem m ab 🡪 Wieso?? Intuitiv müsste Runtime durch steigende Anzahl an NB ebenfalls steigen? Spezielle Gurobi-Funktionalität?

VaryingMaxA:

* Keinerlei Sensitivität
* Vermutlich sensitiver, wenn andere Verteilung(sparameter): Höhere Mittelwerte und Standardabweichungen
* Runtimes ebenfalls konstant
* Avg Simulation Time zackenförmig; bei ungerade >> gerade (Wieso?? 🡪 Code checken wo gerade vs. ungerade getestet wird)

VaryingN:

* Greedy bei kleinstem n=200 mit 100% Performance, da alle Kunden angenommen werden können
* Dyn und One können 100% nicht erreichen, da erste n\*eps Kunden immer abgelehnt
* Sobald zu viele Kunden, deutlich mehr als verfügbare Ressourcen, bleiben Dynamic und One-Time relativ konstant, während Greedy deutlich abnimmt
* Relativ rapide Abnahme aller Performances, sobald nicht mehr de facto jeder Kunde angenommen werden kann
* Interval startet schon auf niedrigerem Niveau, da erste 10% immer abgelehnt
* Amazon Learner scheint wiederum nur bei überschüssigen Ressourcen / wenigen Kunden ähnlich wie andere Benchmarks zu funktionieren
* Weitere Analyse nötig
* Runtimes: Steigen bei allen Benchmarks linear; scheint linearen Zusammenhang zur Anzahl Kunden zu geben; analog steigen auch Avg Simulation und Avg Permutation Time

VaryingNoSim:

* Manuell auszuwerten

VaryingPerm:

* Performances konstant, unabhängig von Anzahl Performances, zumindest bei Standard-Eps von 0.01
* D.h. „Erwartungswert“ über Permutationen neben ist nicht notwendig für valide Ergebnisse
* Runtimes ebenfalls konstant

Dynamic gut bei kleinen Eps, heterogenen Kunden, knappen Ressourcen bzw. vielen Kunden