Master's Thesis: Offer Networks Simulation and Dynamics

ZAR GOERTZEL

August 4, 2017

CONTENTS

1 ABSTRACT

This thesis explores the feasibility of using optimization algorithms to facilitate barter as a market mechanism. A dynamic scale free graph models the variability of tasks, and a uniform acceptance probability p models the likelihood users will be satisfied with suggested exchanges. The ideal exchange size depends on p. This thesis tests an optimal polynomial time algorithm (for $p = 1$), a greedy shortest cycle, GSC, algorithm, a dynamic matching algorithm, and a two-way matching algorithm. In addition, there is a stigmergic approach, HOR, to using the optimal algorithm with $p < 1$, where when a user rejects a match the surrounding users are kept in hold as another match is sought. This approach allows matching to be more effectively done with low p. The greedy shortest cycle algorithm performs almost as well as the optimal algorithm, and far better for low p. In conclusion, GSC with HOR make barter exchange feasible.

Department of Computer Science, University of Copenhagen (DIKU)

2 INTRODUCTION

Barter has historically been infeasible on the large scale, and prior to money various methods of debt tallying were used [1]. One difficulty is that in barter both parties in an exchange must have a coincidence of wants simultaneously; money or debt allow a service to be offered before its equal is received. Another difficulty with barter is that exchanges involving more than two parties will be difficult to organize, whereas money allows someone to sell to one person and buy from another.

Money works well, yet not seamlessly. There are cases where people would rather exchange than use money. For example, by using Intervac [1] [2] homeowners can exchange homes for the holidays, avoiding the hassles of renting their own home out for money. In addition, one could exchange for a home of equal market value; yet with money a home-owner is unlikely to get parity: there are taxes and middle-men involved at each step of the way, and the other home-owner may also be seeking to rent at a profit.

Exchanging used goods may be preferable to selling for profit and using those profits to buy other used goods. Swap It [2] and SwapAce [3], for example, allow buying, selling, or swapping. BookMooch [4] allows users to exchange books via points, essentially a currency. Some people need different sized shoes for each foot and under normal circumstances need to buy two pairs, or an amputee would have to buy two shoes uselessly. Fortunately there is a National Odd Shoe Exchange [5] to help all these individuals save money. Data can also be exchanged. Potentially, even groceries could be bartered locally: you have some extra bread and could offer some to a neighbor, if they only knew.

Tradebank [6] provides barter matching services and has their own currency; however, they use humans to assist with matching rather than automation. Barter Network [7] provides a similar service. These services both show the value of matching users and helping them to find a coincidence of wants, even if private currency is still used. There is significant corporate barter for various purposes: tax reduction, cooperation, handling surplus goods, and even avoiding economic transactions (e.g. Iran bartering oil for goods and assets) [3]. Data can also be exchanged [4], see: DATATANG[8], The Data Exchange[9], and XOR Data Exchange[10].

Goods that cannot be sold for moral reasons [11][6] are also exchanged using optimization algorithms. The best current example of this is kidney

1 www.intervac-homeexchange.com/
2 https://www.swapit.la/
3 http://www.swapace.com
4 http://bookmooch.com/
5 http://oddshoe.org
6 http://tradebank.com
7 http://www.barternetworkinc.com
8 Example from [5]: datatang.com
9 thedataexchange.com
10 xor.exchange
11 What Roth calls *repugnant* transactions.

exchange programs in the United States, Netherlands, Canada, the United Kingdom, Portugal, Australia, and Israel [7], as the sale of kidneys is prohibited in every country but Iran.

There are already good use-cases when the matching and user-interface has to be custom-tailored to the item or service being exchanged. When possible, direct exchange is more efficient. eBay has 100 million items for sale each day [8]. How many of them could be satisfied with an exchange?

The author and collaborators with the Global Brain Instiute [12] have called such a general barter exchange network an Offer Network [9] [10] [11] [12]. The socioeconomic and political implications of a large scale global offer network are discussed in depth. Money, to the extent its still useful, will be grounded transparently in what people are willing to exchange with each other. People in developing countries will more easily enter the global market [13]. In Open Production Netorks [9] production lines can be made transparent and searchable, making anyone skimming off eploitative levels of profit readily apparent. As the marginal cost of daily and infrastructural needs trends towards zero, human exchanges will become dominated by I-You exchanges rather than monetary I-it ones[14]. Internet of Things and 3D printer technology will make personalized designs more important than the materials, which will then be an attention service and more suited to exchange [11]. In fact, AI agents in internet of things technology may want to use an offer network of their own, or, a framework such as the Web of Needs [13] where agents upload there needs. The Web of Needs is design to allow external applications to perform services on agents needs around the globe.

While some instantiation of offer networks can be foreseen in the future, and can be predicted to have numerous subtle and profoundly beneficial impacts, the feasibility of satisfying users via exchange alone needs to be verified. This thesis sets out to test whether optimization algorithms can be used to overcome the second difficulty barter exchange faces: allowing and arranging k-way exchanges ($k > 2$). If enough users can be satisfactorily matched without waiting too long, then offer networks will be feasible without more advanced artificial intelligence methods.

This thesis works with a minimal viable prototype (MVP) of an offer network's core functionality. In this prototype, **users** upload (offer, request) pairs (hereboy called **ORpairs**) of **tasks** (either services or goods). An optimization algorithm is run to find and suggent matches, exchanges with parity; the frequency depends on whether a batch or dynamic matching is being used. The first questions are how many users get satisfactory matches and how long this takes, where time is measured in terms of ORpairs added to the offer network. An upper-bound on performance is the maximum possible matching (with $p = 1$), and the maximum 2-way exchange matching is a lower-bound.

12 https://sites.google.com/site/gbialternative1/

13 This is already happening to some extent with the internet and cryptocoins.

14 I-You interactions are personalized, whereas in I-it interactions one treats the other as an object.

3 RELATED WORK

Apart from the work at the Global Brain Institute, the most extensive collection of work deals with kidney exchanges. Next there are articles and theses in various fields studying barter exchange networks.

3.1 Kidney Exchange

There has been extensive research for kidney exchanges by computer scientists [14][2], econmists [15][16], and doctors [17]. Kidneys are illegal to buy and sell with the exception of Iran [6]. At least a third of patients with *willing* donors cannot get a transplant because of blood type incompatibility or positive crossmatch [17] [15]. Thus starting in the early 21st century, incompatible donors and kidneys have been paired up and entered into national kidney exchanges.

In the offer network terminology, there are four task types: A, B, AB, O, and each user enters an ORpair: (donor type, patient type). Due to the limited number of types, Roth, Sonmez, and Unver [16] show that the maximal matching can be obtained with only up to 4-way exchanges [16] [17] in a thick market; moreover, the only benefit to 4-way exchanges are bounded by the number of rare (O, AB) ORpairs. This is good, as one distinguishing feature of kidney exchanges compared to general offer networks, other than having only four task types, is that an exchange of size k requires $2k$ simultaneous surgeries. Chains can be done sequentially, passing along a donation, to enable 51+ transplants [18]. As the donation is passed along, there is no need for simultaneous surgeries. Another distinguishing feature of kidney exchange: finding the optimal solution is very important as every patient fewer is a potential death. With four task types and tens of thousands of users, the graphs are very dense.

Roth [16] uses patient distributions from U.S. Organ Procurement and Transplantation Network (OPTN) and the Scientific Registry of Transplant Recipients (SRTR) data to randomly generate populations of size 25, 50, and 100. Roth finds, in agreement with the theoretical results, that 4-way (or unbounded) exchanges do not provide much advantage over 2&3-way exchanges.

Prior to this Roth, Somnez, and Unver investigated TTCC (Top Trading Cycle with Chains), an extension of Galel and Shapley's TTC (Top Trading Cycle) algorithm for housing exchange, for kidney exchange [15]. In TTC, each (donor, patient) pair has a list of most compatible kidneys, and points to the top of the list. Cycles are found, matched, and the ORpairs removed; then the poitners for remaining ORpairs are readjusted. These steps are repeated

15 A crossmtach test involves mixing cells and serum to detremine if the patient will reject the donor's kidney.

16 With the exception of tissue type incompatibilities / positve crossmatches.

17 Using a general result that the size of matching needed is bounded by the number of types being exchanged.

until there are no more cycles. In TTCC a cadaver queue is used to enhance the exchange: a donor who donates a kidney to the waitlist gets a priority position in the queue. This allows chain formation ending in an ORpair willing to accept a good queue position, which is still a lottery. The chain-selection rule of starting with the highest priority pair and keeping the chain (in case it grows in later rounds) results in a pareto optimal and strategy proof matching [18]. Roth experimented with simulations of population size 30, 100, and 300, finding TTCC does help. However, the longest cycles were of length 26, and chains of length 15.

Biró, Manlove, and Rizzi in *Maximum Weight Cycle Packing in Directed Graphs, with Application to Kidney Exchange Programs* [14] study the kidney exchange problem from a computer science perspective. Formally, the problem is to compute a **cycle packing of maximum size or weight**. The problem of finding the maximum with only 2&3-cycles is APX-complete [19]. This result also applies to (<k)-cycles for $k > 2$, cycles of any bounded length, and to maximum cycle packings based on cycle weight. The case of 2-cycles and maximum cycle packings based on edge-weight alone have polynomial time solutions. Biró describe, but don't use, a $(k-1+\epsilon)$-approximation algorithm using augmenting path search, and an exact algorithm that runs in $\mathcal{O}^*(2^s)$ [20] where $s < |edges|/2$ is the size of a set, S that covers at least one edge from each 3-cycle in the graph. The algorithm uses maximum matching on a graph constructed using subsets of S, thus the exponential.

Biró use their exact algorithm to exchange kidneys with NHS Blood and Transplant (NHSBT) in the UK. The ($\leqslant 3$)-exchanges performed twice as well as the 2-exchanges, and 0.79 that of the unbounded exchanges. In practice, NHSBT chose to use the $\leqslant$ 3-exchanges as a reference while choosing a slightly smaller exchange with more 2-exchanges to diminish complications and the chance of the kidney exchange not happening for some reason.

Abraham, Blum, and Sandholm [2] develop an algorithm based on integer linear programming (ILP) that is used by the *Alliance for Paired Donations*, a leading kidney exchange in the US. Non-incremental ILP methods can easily have billions of constraints and run out of memory, as either the variable or constraint number is exponential in input size. Thus Abraham use incremental problem formulation methods: constraint generation and column generation. Abraham finds column generation with a cycle-based ILP formulation to provide optimal exchanges fast enough for use with over 10,000 patients. In the branch and price method used a depth first search is used to look for positive price cycles. A further advantage of ILP is that additional features are easy to add as constraints.

Anderson et. al [19] develop an efficient constraint generation approach modeled on the prize-collecting traveling salesman praoblem (TSP) that also handles chains well.

18 That is, no other matching is strictly better for an ORpair without being worse for another, and lying about preferences does not confer any advantage.

19 And thus NP-complete.

20 Parametrized complexity: $\mathcal{O}^*(2^s) = \mathcal{O}(2^s f(n))$ where $f(n)$ is a polynomial in the input size, n.

Glorie, Klundert, and Wagelman [20] improve upon Abraham [2] by transforming the positive cycle search into one for negative cycles and then uset he Bellman-Ford algorithm to find them in PTIME instead of exponential time in the cycle or chain size limit.

Plaut, Dickerson, and Sandholm [21] notice an error in Glorie's algorithm [20] in not handling loops in the Bellman-Ford algorithm properly. Fixing this adds a factor of K (the chain size limit)[21]. This algorithm significantly outperforms the previous state-of-the-art methods, with the exception of Anderson's TSP approach [19] on unbounded chains, which are less likely to be fully realized in practice. An interesting point to note is that the algorithm performs significantly better with more altruistic donors.

Jia, Tang, Wang, and Zhang [22] prove the existence of PTIME approximation algorithms for up to (k)-way exchanges based on the k-SET-PACKING problem. They test a local search approach as in [23]. One heuristic to initialise the local search is to greedily add cycles from vertices in sorted order. One ordering is Product of Degrees[22], like Maximal [23]. The other ordering is to solve an LP relaxation and use the resulting variables. The local search achieves 90% of the optimal performance under all settings, and 98%+ with the LP relaxation heuristic using only a small sample of the 3-cycles in the graph. Chains are dealt with greedily before running the cycle cover algorithms.

Over the course of his PhD, *A Unified Approach to Dynamic Matching and Barter Exchange*, John P. Dickerson [7] does an impressive amount of innovative research. Theoretically, there is a high probability an optimal matching with cycle and chain length capped at 3 in a dense graph; moreover, long chains do not add much benefit due to the probability of rejection due to positive crossmatch testing. Position-indexed formulations (PICEF) for kidney exchange [7] [24] are developed that perform reliably faster than other models in practice; alas, the linear programming relaxation is less tight. Smaller representations of the kidney compatibility graph allow for an PTIME algorithm[23], as well as allowing for SAT methods to be used [7] [25]. In a static model, multiple rounds of rematching rejected cyles results in 75%+ more matches.

One topic in [7] failure-aware matching: clearing the market while taking failure probability $p_{u,v}$ into account; however to solve the pricing problem in PTIME, failure probability must be assumed uniform. Accounting for p with high probability gives a matching with linearly higher expected value than the maximum cardinality matching [7] [26]. Bimodal matching with p_{low} and p_{high} worked significantly better than only p_{avg}. Edge-weights to prioritize hard-to-match highly sensitized patients for fairness to prevent marginalization works and does not decrease overall performance much.

21 Apparently Dickerson has found an error in this fix for chains and in his PhD thesis [7] proves that deciding whether a negative chain exists or not is NP-complete. See: http://jpdickerson.com/pubs/plaut16hardness.pdf

22 $(d_{in}(v)+1)*(d_{out}(v)+1)$

23 The algorithm is exponential in number of types, but that is fixed in kidney exchange.

Pre-testing a random sample of edges for compatibility also significantly improves performance.

Another topic in [7] is less myopic dynamic matching[24] by learning vertex/edge/cycle/graph potentials. Thus, for example, a chain of 2 patients may be held back in one round due to its potential to form a 3-cycle in the next round. SMAC (sequential model-based optimization for general algorithm configuration) [27] is used to determine potentials. $10-20\%$ gain seems achievable.

Finally, Dickerson [7] [28] look at multi-organ exchange. There are far fewer lung-donors; however a combined market causes a linear increase in the number of exchanges due to altruistic "chains that thread through the liver pool."

3.1.1 *Offer Network*

In *A Dynamic Model of Barter Exchange*, Anderson, Ashlagi, Gamarnik, and Kanoria [29] investigate when exchanges should be performed in an offer network [25] with 2-way, 2&3-way, and chain based matching. A **chain** is a match started by a gift, which Anderson calls the **head**. The chain ends when a user whose ORpair receives a gift has no-one to pass the gift on to; this user then becomes the **bridge** to initiate a chain in a new timestep. An Erdos-Renyi graph model [30] is used with probaiblity p of an edge between any two nodes. Unlike in this thesis, each user has a unique task, so a task is removed once it is matched.

The experiments use a simulation with 16,000 arriving nodes, $p \in [0.04, 0.1]$, and for batch sizes 2^m for $m \in [0, 6]$. Anderson finds experimentally and theoretically that greedy matching is optimal, i.e., match cycles/chains as soon as they are found.

The average waiting times are:

2-way	$\ln(2)/p^2 + o(1/p^2)$
2&3-way	$\mathcal{O}(1/p^{3/2})$
k-way (hypothesized)	$\mathcal{O}(1/p^{\frac{k}{k-1}})$
chains	$\mathcal{O}(1/p)$

An exception to the result in Anderson [29] is found in *Matching Market Design* by Akbarpour [31]: if users take ORpairs off the market and the time this will be done is known in advance, then a patient algorithm that waits until the time step before an ORpair is taken off the market to match it performs better than a greedy algorithm. On the other hand, if the time the ORpair leaves the market is unknown, greedy performs better.

In *On Efficient Recommendations for Online Exchange Markets*, Abbassi and Lakshmanan [23] perform experiments similar to those in this thesis. Abbassi uses a model where each user has an **item list** and a **wish list** and the system

24 Like greedy algorithms with respect to a dynamic model. Myopic optimization does the best possible in the current time step ignoring future time steps.

25 barter exchange

tries to recommend the maximum exchange, assuming the user is willing to ecxhange any wish list item for any possessed item [26] A probabilistic model that can model user/item preferences or reputation is also analyzed, but not experimented with. The model is scale free with respect to item popularity, with slight deviation in wish list and item list popularity. Three algorithms are experimented with:

- Maximal: find a greedy edge-disjoint cycle cover M times, and use the one with maximum weight.
- Greedy: Greedily add cycle with largest weight.
- Local Search: For each cycle, check if adding it (possibly displacing some cycles) increases the cover weight.

Performance is measured in **coverage**, how many items are exchanged. Maximal performs almost as well as the slower algorithms[27], and Abbassi finsd Maximal's performance increases significantly until $M \approx 100$.

Next, in *Fair Recommendations for Online Barter Exchange Networks* Abassi, Lakshmanan, and Xie [32] test asynchronous exchange via a credit point system (modeled on Yahoo! Answers points more than money). Abassi find 50+ time slots are needed for synchronous exchanges as in [23] to facilitate as many transactions as the asynchronous model. This difference seems similar to that between the wait-time for (gift-headed) chains and and cycles in [29], which also makes sense as all users start with an initial credit amount that can be used to get an item without giving an item, i.e., to get a gift[28]. Four optimzation problems are formed: maximum value gained (based on credits), maximu cardinality exchange with maximum value gained, minimum transaction cost (in addition to the previous ones), and MAX-FAIR, maximum fairness, (minimizing the maximum credit gained by any user). All are PTIME computible except for MAX-FAIR. A greedy heuristic and an LP rounding heuristic are tested for MAX-FAIR: greedy achievse a factor of 2.3 to the optimum and the LP rounding achieves close performance. Abassi also obtains a data snapshot from `ReadItSwapIt.co.uk` and verify a scale free distribution of item popularity.

Rappaz, Vladarean, McAuley, and Catasta use task similarity to provide ordered lists of reccommended swaps in *Bartering Books to Beers: a Recommender System for Exchange Platforms* [33]. Rappaz crawls the matching sites Bookmooch[29], Ratebeer[30], `/r/gameswap`[31] to get real datasets [32]. Rappaz note that with the exception of Ratebeer, less than 5% have at least one eligible swapping partner, and that on Bookmooch approximately 33% of items

26 If each user has one item and wishes for one item, then Abbasi's model is the same as in this thesis.

27 Let V be the vertices, E edges, and $\mathcal{B}$ the cycle cover. Then Maximal runs in $\mathcal{O}((|V| + |E|)|\mathcal{B}|)$ and Greedy in $\mathcal{O}(|V|^{2k})$.

28 Investigation into the relation of initial credit distribution and effectiveness of the credit mechanism would be interesting.

29 `bookmooch.com`

30 `ratebeer.com`

31 `reddit.com/r/gameswap`

32 Available at `swapit.github.io`

transacted were not in the user's wish list prior to the transaction. This is the motivation for developing a matrix factorivation based recommender system that gauges users' preferences based on their lists rather than only using swaps. Rappaz also incorporates social bias (whether users have checked each other out or interacted). Rappaz also decides to accept conflicting recommendations as acceptance probability is not high.

3.1.2 *Miscellaneous*

Fang, Tang, and Zuo study *Digital Good Exchange* [5] where, as with data, goods are not deplenished when exchanged but their value decreases. While in genreal, finding a non-trivial, PNE (pure nash equilibrium) is NP-complete, the problem can be solved in PTIME when agents have unit demand, i.e., request one other agent's data at a time (which may be reasonable in practice). The unit demand case fits into the basic offer network framework: (offer data d_i, request data d_j). An encouraging case for open data[33] is that each agent asking for every other agent's data is a PNE weakly dominating all other strategies (except asking for nothing, complet secrecy).

Cooper and Garcia-Molina [34] design and test two protocol for trading data to replicate it; unlike in Fang [5], the more sites with the data the better. Their protocols focus on trading collection replication or trading deeds to data on their sites. The trading deeds algorithm reaches high global reliability faster. This service could perhaps be framed in an offer network framework.

Anagnostakis and Greenwald [35] test exchange based methods for peer-to-peer file sharing as an alternative to two-way credit systems (upload and download). They get positvie results for up to 5-way exchanges. Exchanges are cycles detected by peers that maintain an incoming request tree (only up to depth 4). No effort is made to find optimal cycle covers in the decentralized system.

Cabinallas [36] [37] has done a lot of theoretical work on peer-to-peer bartering[34] in papers and his PhD thesis. Cabinallas compares 2-way, 2&3-way, and optimal [35] exchanges. Cabinallas' primary focus is on distributed bartering via selfish agents without global knowledge. As in Abbassi [23] [32] agents have item and wish lists, but an Erdos-Renyi model is used for the item distribution. The level of satisfaction obtained over time in fully-connected to sparse graphs is tested: the market was found sterile when not the average degree was less than 11. Kyle MacDonald, through a series of exchanges, traded a red paper clip for a house [36]. Cabinallas runs simulations to test whether this is possible in general when agents have personal valuations of items apart from a set market price: the answer is yes. The main use-case explored is distributed barter based directory services for use with DNS, community-based replication, or other content distribution

33 The position that data should be freely available to everyone.
34 decentralized bartering
35 via Munkres algorithm
36 http://oneredpaperclip.blogspot.dk/

networks to provide more scalability and reliability than server-based implementations. The model is a network of directory nodes clients can query (that may then query each other).

4 STATE OF OFFER NETWORK RESEARCH

Most work in the offer network field myopically focuses on kidney exchange. While there are general benefits to the research, there are also limitations:

- There are only 4 types of items (kidneys).
- There is mainly interest in optimal solutions. Ironically, a moderately more accuarte model can lead to more matches than optimal solutions. This can be seen looking at the gains from accounting for acceptance probability [7] [26] and the performance of near-optimal approximation algorithms [22].
- Primarily Erdos-Renyi graphs are analysed, both due to similarity to kidney exchange graphs[37] and for analytical simplicity.
- Domain-specific techniques are used to gain performance with ILP[38] approaches [7] [24] [25] [20] [19] [39].

Cabinallas' work [36] shows some promise for decentralized offer networks, but uses a uniform model as in the kidney cases, and spends more time discussing models than testing their generic feasibility.

Abbassi [23] [32] use a scale-free graph[40] (and get data verifying this model) and initially test Maximal (greedy cycle cover) with two algorithms with known approximation bounds. Next they compare synchronous barter to a credit-based system, with dismal results. While acceptance probabiltiy is mentioned, no experiments are done; nor is marginalization of users handling unpopular tasks dealt with in the scale-free model.

5 CONTRIBUTIONS

First this thesis replicates Abbassi's experiments [23] with a theoretically motivated variant of Maximal, GSC (Greedy Shortest Cycle), in a graph model with a scale-free item distribution, and compare it to the maximum weight[41] cycle cover, and a 2-cycle cover.

Next this thesis tests the importance of matching frequency in a dynamic offer network setting, similar to Anderson's work with Erdos-Renyi graphs

37 Kidney type distribution is not uniform, and some dicrepencies between theory and experiment have been observed [7].
38 Integer Linear Programming
39 In practice, an offer network should provide an API for specialized matching agents.
40 item distribution
41 cardinality if weigths are uniform

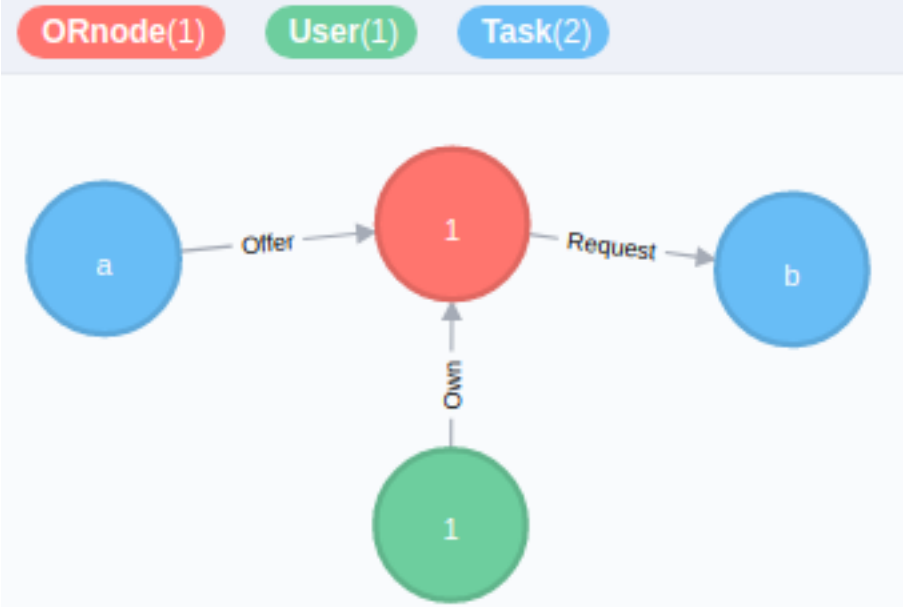


Figure 1: User 1 uploads an ORpair to offer task a and requests task b in exchange.

[29]. Also included is a dynamic matching method that tries to match only new ORpairs every 1-3 added.

Uniform acceptance probabilities p are tested and investigated analytically: the chosen heuristic depends on the range of p. Normally a long cycle with one rejecting edge has to be rejected, which is why short cycles are generally sought. A heuristic to salvage such a cycle's feasibility is tested.

In the ReadItSwapIt data from Abbassi [32] 90% of items were only requested by 1 user. Thus settings that marginalize such users least are investigated[42].

6 MODEL

An **offer network** instance can be formally modeled as a directed graph, G, where vertices $task \in T$ represent tasks, $user \in U$ users, and labeled directed edges $(task_a, task_b)$ represent an offer by user $user_1$ to do task $task_a$ in exchange requested task $task_b$. Due to implemention details in Neo4j[43], (offer, request) nodes are created for labeled directed edges, called **ORpairs**. This is the **the task-centric model**[44]. See Figure 1.

A matching of users who can satisfy each other is a vertex cycle. See the simplest case below in Figure 2. There is a match acceptance probability p, which is assumed to be uniform and independent[45].

42 Graph analytics are necessary as an item requested but not offered cannot be matched.

43 https://neo4j.com/

44 An alternative is an **ORpair-centric model** where each ORpair is directly linked to all offer or request compatible ORpairs.

45 Moreover, Dickerson [7] notes this assumption potentially also protect users, such as badly sick patients, from being marginalized in addition to being simpler to analyze.

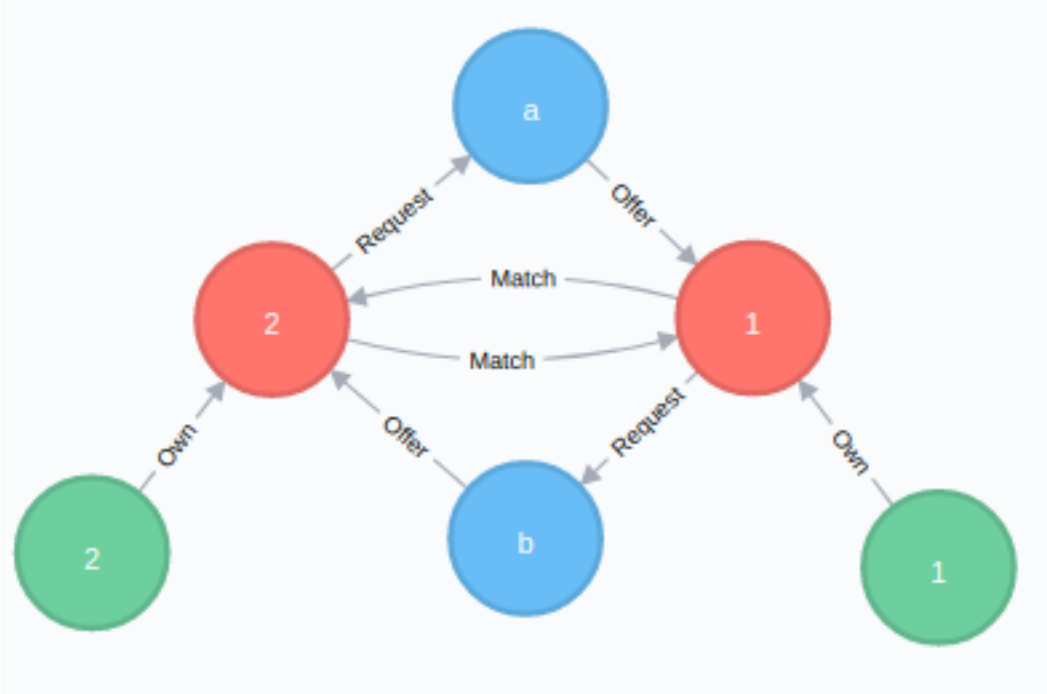


Figure 2: User 2 uploads an ORpair to offer task b and requests task a in exchange. Now there is a cycle/match.

6.1 Scale-Free Task Distribution

The ReadItSwapIt, Bookmooch, Ratebeer, and Reddit Gameswap data is scale-free. This means that for large degrees k, the fraction of nodes with degree k is proportional to $k^{-\gamma}$[46]. γ is called the power law exponent. In colloquial terms, this means that a few tasks are very popular and most tasks are only offered or requested by a few users. There will likely be a small difference in *offer* and *request* popularity for tasks.

Worth mentioning on this note is a user-centric study of the eBay graph [8][47]. They noted the following properties on eBay:

- Skewed degree distribution: i.e., there are a few big sellers.
- Dissasortativity: users tend to buy and sell different types of products.
- Linear preferential attachment in terms of positive feedback.
  Strong aversion avoidance of users with negative feedback.
- Densification over time.
- No rich club connectivity (basically because big nodes are mass sellers).

Dissasortativity implies that there will be little clustering: there will be many ORpairs between diverse task types[48].

46 https://en.wikipedia.org/wiki/Scale-free_network
47 The data is obtained via crawling eBay feedback.
48 The specialization of many exchange sites and services may seem to contradict this; however this may also be due to the ease of developing specialized mechanisms.

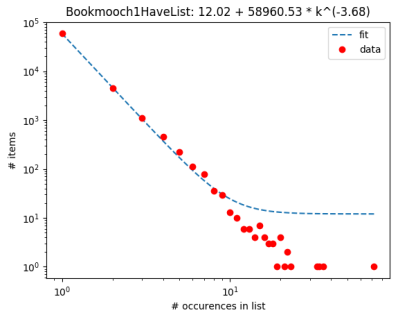


Figure 3: Bookmooch data over 1 year: have (offer) list

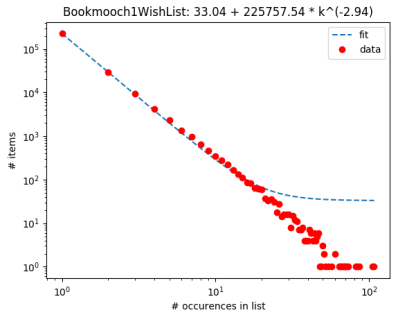


Figure 4: Bookmoch data over 1 year: wish (request) list

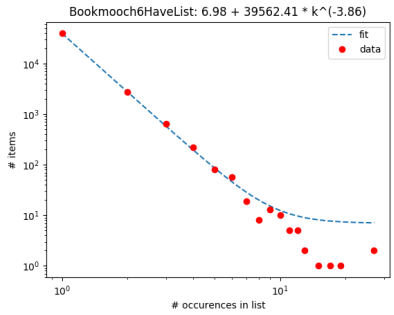


Figure 5: Bookmooch data over 6 months: have (offer) list

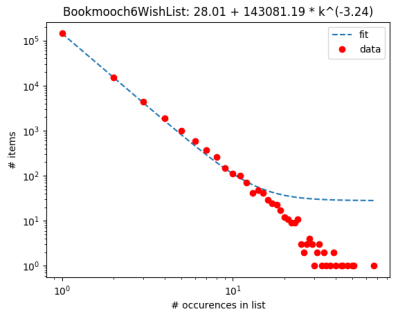


Figure 6: Bookmoch data over 6 months: wish (request) list

Densification over time implies new ORpairs are added at a faster rate than new tasks.

6.1.1 *Data Analysis*

In this section the task popularity distributions of the datasets provided by Rappaz [33] are analyzed. The offer and request task distributions are similar; however the request task distributions have lower power-law exponents. That is, there are both more requests and more requests in many users' wish lists.

Note that an task in a wish list can potentially be part of an exchange involving any task in a have list. This means that, if the average wish list size is 3.4 and have list size is 5.4, as in the ReadItSwapIt data [32], a task in one wish list (degree one) may have in the model in this thesis degree 5. This effectively makes the power-law exponent lower for small degrees (only asymptotically the same).

6.1.2 *Scale-Free Graph Model*

The goal is to generate a graph with a power law distribution with a small difference in the in and out degree distributions. To this end a generator for directed scale-free graphs is modified. The modified graph generator

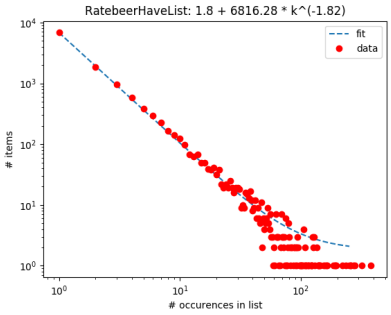


Figure 7: Ratebeer data: have (offer) list

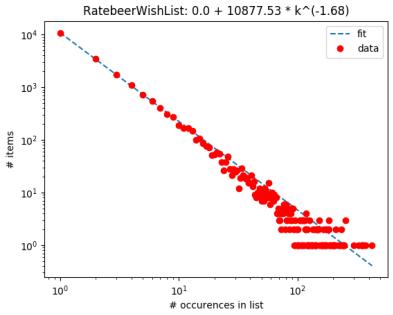


Figure 8: Ratebeer data: wish (request) list

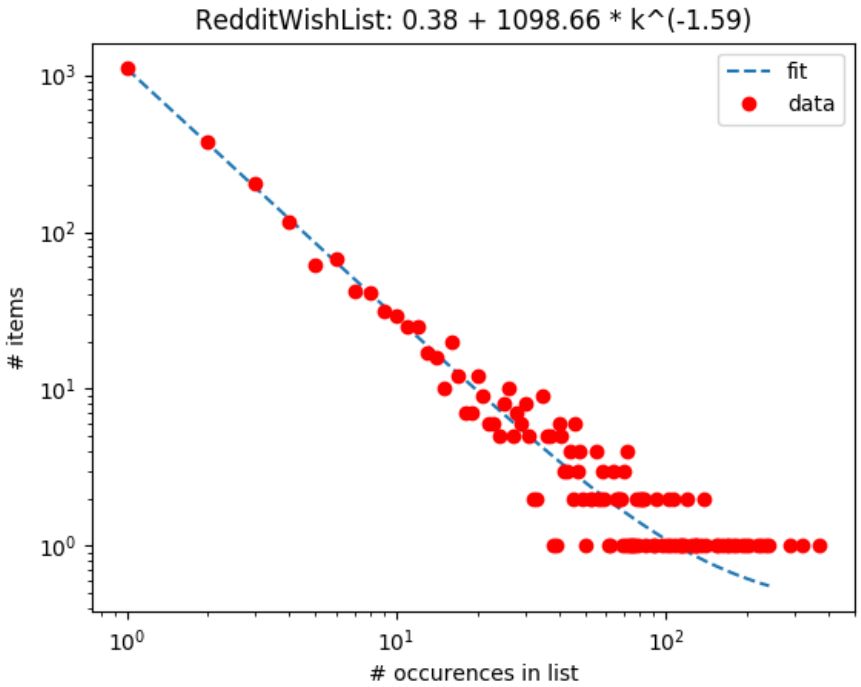


Figure 9: Reddit data: wish (request) list

from NetworkX [38] is described by Bollobás, Borgs, Chayes, and Riordan in *Directed Scale-Free Graphs* [39]. It's a preferential attachment model.

The graph starts with a directed triangle.

Each time an edge[49] is added with probability p_d the directed scale-free graph generation is used that chooses nodes based on in/out-degree, otherwise node choice is based on total degree.

- α, create new node[50] v and choose w based on in-degree distribution.
- γ, create new node w and choose v based on out_degree distribution.
- β, choose v based on out-degree distribution and w based on in-degree distribution.

Then add (v, w) to the graph.

There are parameters δ_{in} and δ_{out} added to the degree so that nodes with zero degree are still chosen with non-zero probability (and $\frac{\delta_{in}+\delta_{out}}{2}$ in the total degree case). Let

$$c_1 = \frac{\alpha+\beta}{1+\delta_{in}(\alpha+\gamma)} \text{ and } c_2 = \frac{\beta+\gamma}{1+\delta_{out}(\alpha+\gamma)}$$

$$X_{IN} = 1 + 1/c_1 + c_2/c_1(\delta_{out} + 1_{\{\gamma\delta_{out}=0\}})$$

$$X_{OUT} = 1 + 1/c_2 + c_1/c_2(\delta_{in} + 1_{\{\alpha\delta_{out}=0\}})$$

X_{IN} and X_{OUT} are the power law exponents. Note that in this model in-degree and out-degree distributions are independent. The effect of a high p_d (for similar offer/request task popularity) is not immediately theoretically clear.

The modification of the model generates a set number of edges and for each edge adds a new task with probability $\alpha+\gamma$.

Note that $0 < c_i \leqslant 1$ and $1 < X_{IN}$. If $\delta_{in} = \delta_{out} = 0$ and $\alpha = \gamma$, then $X_{IN} < 4$. Generally, for one exponent to be large the other has to be smaller.

In this thesis the distribution is verified empirically in the same way as those in real datasets.

6.2 Dynamics

The dynamic offer network experiment starts with a scale free graph generated as above with $N_{initial}$ ORpairs. Each step 1 ORpair is added. There are n^u ORpairs for each user, so every n^u steps a user is added. With probability $\alpha+\gamma$ a new task is added with the ORpair. The experiment is run until N_{end} ORpairs have been added. Every n_{match} steps matchings are suggested, accepted or rejected by users, and accepted ORpairs are removed from the graph.

49 ORpair
50 task

It is possible to make n^u be probabilistic as with new tasks, and it's possible to make n_{match} depend on the graph size.

The series of graph updates from the start to completion N_{end} are pre-generated so that all algorithms can be tested on the same dynamic graph. Note that the graph used to determine the scale-free task distribution (in NetworkX) is separate from the offer network graph (in Neo4k) that the matching is done on. This is to prevent ORpairs being matchend and removed from altering the task distribution[51]

6.3 Cycle Size and Acceptance Probability

6.3.1 *Motivation*

The optimal solution clearly depends on acceptance probability, primarily biased by the optimal cycle size. In general, finding a maximum edge-disjoint cycle cover with cycle-weights (to account for acceptance probability) is NP-hard [14]. Dickerson [7] [26] find that integer linear programming with a model that optimizes for expected gain based on a uniform or bi-modal (high and low) acceptance probability performs linearly better[52].

When using heuristics, however, one possibility is to simply choose a different algorithm depending on the currest estimation for p. Included below is some basic probabilistic guidance.

6.3.2 *Analysis*

In a cycle c of size k, the probability of acceptance is $p_k = p^k$.

Thus the expected number of satisfied users is: $\mathbf{E}[c_k] = kp^k$. For a given p, the best cycle size is given by $k = -\frac{1}{\ln p}$, and the best attainable for p is $-\frac{1}{e \ln p}$.

As can be seen in Figure 10, the optimal cycle size exponentially incresaes above $p = 0.95$ ($k \approx 19$), so the maximum without cycle-length constraints will likely suffice[53]. And in Figures 11 & 12 one sees that 2-cycles are optimal for low-p, $p < 0.7$. The PTIME heuristic is less clear for the following cases:

- mid-p: $0.7 < p < 0.85$ where $2 < k < 10$ is optimal.
- high-p: $0.85 < p < 0.95$ where $10 < k < 30$ is optimal, yet the larger cycles are little bitter than shorter ones.

51 In reality, there will be a relation between these; however, modeling this relation is beyond the scope of the thesis. Futhermore, more matches of a certain task are likely to increase the task popularity.

52 A performs linearly better than B means $\mathrm{Utility}(A, n) = \mathrm{Utility}(B, n) + \Omega(n)$.

53 Unless the algorithm retruns 1000+ length cycles.

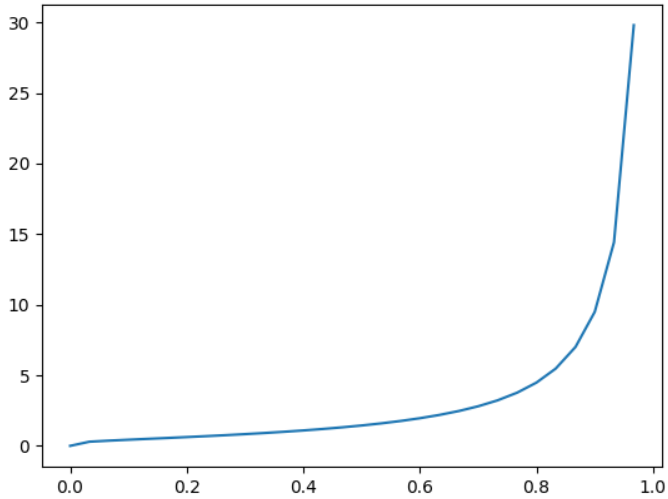


Figure 10: Best k-cycle for p from 0.8 to 1

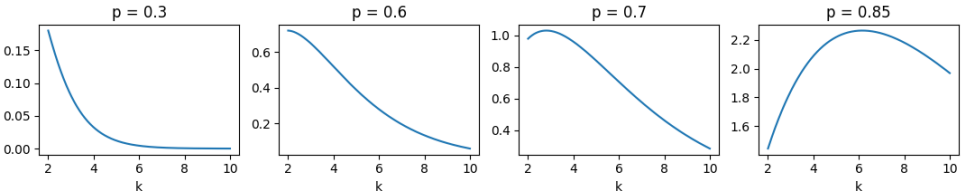


Figure 11: $\mathbf{E}[c_k]$ for $k = 2 \ldots 10$

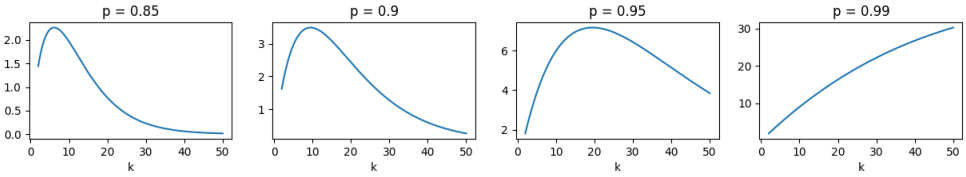


Figure 12: $\mathbf{E}[c_k]$ for $k = 2 \ldots 50$

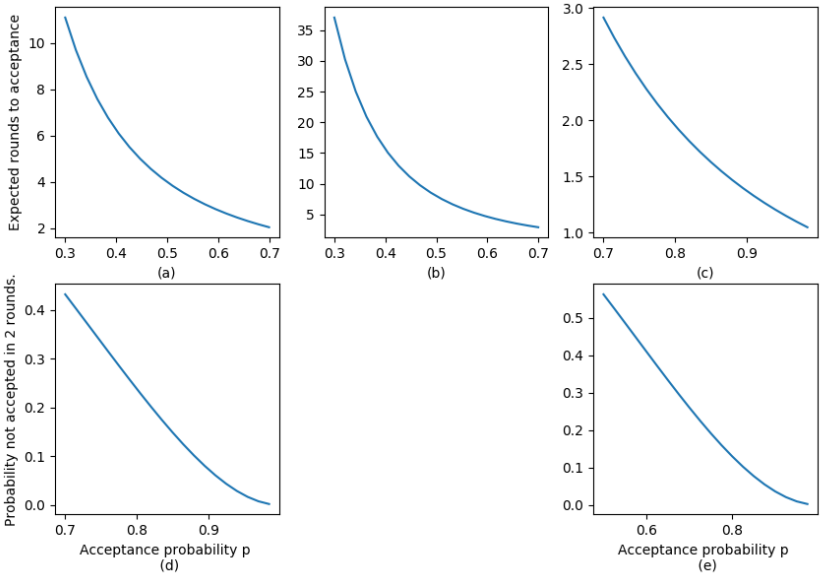


Figure 13: (a)-(c) plot expected number of rounds for acceptance of an ORpair for different p. (a) is for 2-cyles. (b) and (c) for k-cycles. (d) and (e) plot the probability of an ORpair not being accepted in 2 rounds. (d) is for k-cycles and (e) is for 2-cycles.

For human users without very specific task formulations, p likely lies in this range. However, Dickerson [7] [26] found that with kidney exchanges, $p < 0.3$.

6.3.3 *How many matches are needed to for acceptance?*

This thesis tests a heuristic to make long-cycles feasible even without high p. The heuristic satisfies an ORpair if both of its neighbors in the cycle also accept the suggested match (rather than if all users in the cycle accept). If one of the neighbors does not accept the match, then the user is rematched in a way explained in the Methods seciton. The feasibility of this approach depends on how many times a user can expect to be rematched.

In a given round, the probability the suggested match goes through is p^3 as both neighbors in a cycle also need to accept. Thus the expected number of rounds for acceptance follows the geometric distribution: $\sum_{k=0}^{\infty} kp^3(1-p^3)^{k-1} = \frac{1}{p^3}$.

As one can see in Figure 13, one expects 5 rounds to be needed by $p = 0.6$. Fortunately, for $p > 0.7$, one expcets less than 3 rounds to be needed. With 2-cycles, one only needs 3 or 4 rounds for $p = 0.6$ or 0.5. For lower p, the offer network may become frustrating for human users: something will have to be done to increase acceptance probability. For example, matching may have to be done on more detailed task specifications [54].

54 Acceptance probaility is likely proportional to how detailed task specifications are; however, there are also limits as to how detailed humans can be. The reasoning is that when a user

One can also easily calculate the probability acceptance for an ORpair will take more than two rounds: $1-(p^3+p^3*(1-p^3))$.

6.3.4 *Application to Kidney Exchange*

When exchanging kidneys there is a type of incompatibility not checked prior to initial matching: PRA (percent reactive antibody). Roth et al. [16] split the distribution into low, medium, and high PRA with respective 5, 45, and 90 percent probability of positive crossmatch (rejection). The frequency among kidney donors is 70, 20, and 10 percent. Thus for 10 percent of matches, anything but 2-cycle has very low expectation, and for another 20 percent, 2-cycle is still expected to perform twice as well as 4-cycle matches. For the remaining 70 percent, however, 20-cycle matches are best, and 50-cycle still preferable to anything less than 10-cycle matches.

Thus when mixing low, medium, and high PRA the result that 4-cycle matches don't help much makes sense [16].

Morevore, Dickerson [7] [26] calculates acceptance probabiltiy[55] to be at most 30 percent: that is, 2-cycle matches are always preferable to 3-cycle.

7 MATCHING ALGORITHMS

The matching problem is solved on the static graph every n_{match} steps. One goal for the optimization algorithms is to find the maximum edge-disjoint cycle covering, which can be solved as Min-D-DCC is solved [40] [14]. The section 7.1 discusses this PTIME algorithm. This is optimal for $p = 1$.

With $p < 1$, the matching problem is to find the maximum cycle-weight edge-disjoint cycle covering, where cycle weight is $\mathbf{E}[c_k] = kp^k$. This problem is NP-Hard and APX-Complete [14]. See Dickerson [7] [26] for the state-of-the-art ILP (integer linear programming) solving this problem for kidney exchange.

An alternate approximation is to find the maximum edge-disjoint covering with cycles of length $\leqslant k$. The complexity is, however, identical. Biró [14] provide an exact graph algorithm to solve this. There are many good ILP techniques providing exact solutions as well. See [19] [20] [21] [24]. ILP algorithms are easier to add new features to.

rejects a match it is probably because of an unspecified criteria not met. An example in the kidney case is crossmatch testing: 70% of post-match rejections are because of positive crossmatch that is expensive to test. Dickerson [7] find that even a random sapmle of crossmatch tests, that is more detailed specifications, allow for many more matches.

55 Optimistically based on only positive crossmatch.

7.1 Maximum Edge-Weight Matching (MAX-WEIGHT)

The standard algorithm to find a maximum edge-weight edge-disjoint cycle covering in PTIME transforms the graph into a bipartite graph and then finds a perfect weighted matching [14]. The

The bipartite graph has one side for Offers and one for Requests. For each ORpair, (offer=$task_a$, request=$task_b$), create two nodes with edge weight 0. Next an edge is created between the offer node and every request node for $task_a$ with weight > 0. Then a maximum weight perfect matching algorithm can be used. A positive weighted edge for an offer can only be matched if the corresponding request has a positive weighted edge match; if the ORpair is in no cycle, then the 0 weight edge is mached.

The bipartite graph has $2 * |ORpairs|$ nodes and $\mathcal{O}(|ORpairs|^2)$ edges. The perfect weighted matching can be found via Munrkes algorithm or the Hungarian algorithm in $\mathcal{O}(N^3)$ time.

The positive weight can be used to assign preferences to some ORpairs over others, perhaps due to user preferences (soft constraints), proportional to task popularity (degree), or according to reputation. Weights could also be used to represent non-uniform acceptance probability, which may be a useful heuristic.

The drawback of this algorithm is that long-cycles are possible, and in a dense graph, such as for kidney exchange, very likely.

7.2 Maximum 2-way Matching (MAX-2)

In the case that $p < 0.7$, a simple 2-cycle cover will likely perform best. Also $\mathcal{O}(n^3)$.

This can be followed up by another algorithm, such as MAX-WEIGHT.

7.3 Greedy Shortest Cycle (GSC)

A simple heuristic is to pick an arbitrary node, find the shortest cycle[56], and then mark the nodes as matched. Repeat this process until there are no more cycles in unmatched nodes.

The rationale is that for $p < 0.85$ cycles of length $k < 10$ are optimal, and specifying an exact cycle length cap may not provide much additional gain.

The run-time is $\mathcal{O}((|ORpairs| + |tasks| \ln |tasks|)|ORpairs|)$; however, many the searches for many ORpairs will be cut short because of already matched cycles. The search can also be done as part of a graph traversal.

56 For example, using Dijkstra's algorithm

7.3.1 *Relation to Maximal and Greedy (GSC-M)*

GSC is similar to Maximal and Greedy in Abbassi [23]. However, Maximal chooses any cycle and Greedy finds the maximum weight cycle, which is slow. As with Maximal, running GSC M times and choosing the best cover may result in significantly better results.

7.3.2 *Product of Degrees Order (GSC-POD)*

Jia [22] find that ordering the nodes prior to running a greedy shortest cycle cover algorithm boosts results. This seems wise with GSC as well. The reasoning is that a cycle for an ORpair with a well-connected, popular, task may use up the only cycle for an unpopular task. An additional perk of finding the shortest cycle is that this cycle is less likely to use up an ORpair needed by another ORpair unpopular task.

In this thesis task popularity is measured by degree, and the PoD of an ORpair by the product: $deg(offer) * deg(request)$.

7.3.3 *Neo4j Query Implementation of GSC*

GSC can actually be implemented in one query in Neo4j's query-language Cypher:

```
MATCH (o:ORnode)-[reqR:Request]->(req:Task),
p = shortestPath((req)-[link*]->(o))
WHERE NOT exists(reqR.matched) AND
ALL (r IN relationships(p) WHERE NOT exists(r.matched))
FOREACH (r IN link | SET r.matched = TRUE)
SET  reqR.matched = TRUE
WITH FILTER(ornode IN nodes(p) WHERE ornode:ORnode) AS p
UNWIND p as off
MATCH (off)<-[]-()<-[]-(req:ORnode)
WHERE req IN p AND off.offer = req.request
CREATE (off)-[:Match]->(req)
```

7.4 Dynamic Shortest Cycle (DYN-SC)

Dynamically find shortest cycles involving just added ORpairs every few (1 to 10) steps.

DYN-SC is very similar to GSC. DYN-SC is faster but may miss cycles involving matched ORpairs that did not accept their matches, unless these are also included.

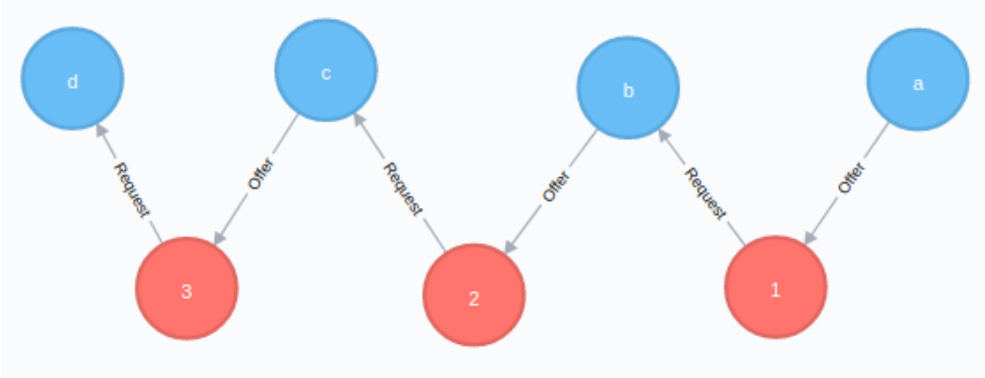


Figure 14: 1,2,3 are ORpairs and a,b,c,d are task nodes.

7.5 Hanging ORpairs (HOR)

In the mid-p, $0.7 < p < 0.95$, range, long cycles will be rejected with high probability. This MAX-WEIGHT infeasible. The Hanging ORpair approach seems to salvage a long cycle with only a few rejectors.

The Hanging ORpair approach is best exemplified by the following two acceptance scenarios for the case in Figure 14 where ORpairs 1, 2 and 3 are part of a larger cycle:

- Users 1, 2, and 3 accept the match: consider ORpair 2 satisfied. User 2 gives $task_b$ to user 1 and gets $task_c$ from user 3.
- Users 1 and 3 accept, but user 2 rejects the match: consider an ORpair like 2 as all that's needed to fix the match. Create a new, hanging, ORpair for users 1 and 3: (offer: $task_c$, request: $task_b$), the converse of ORpair 2.

The core idea is that a match being found implies parity among the tasks involved $(a, b, c, d, \ldots)$. Thus odds are there will be more matches for the new ORpair. Gambling on this, long cycles and 3-cycle coverings can expect the same number of satisfied ORpairs.

Algorithmically speaking, MAX-WEIGHT can be used in the mid-p range to find the maximum weight edge-disjoint cycle cover, and rejectors and hanging ORpairs can be rematched. This is preferable to other methods because there will still be as many rejectors (as p is uniform).

Finally, when multiple adjacent users in a match reject, the acceptors on either side of the streak can be joined to form a hanging ORpair.

7.5.1 *Theoretical Drawbacks*

Some theoretical drawbacks concern the fact that the parity assumption is not always generalizable.

- Subjective value of tasks and willingness to trade for other tasks varies. A hanging ORpair replacing an ORpair of a user with unusual subjective value of a task may be harder to rematch. The example given in

Cabinallas [36] of Kyle MacDonald trading a paper clip for a house demonstrates this effect.

- A task in the hanging ORpair could also be very rare or unpopular, resulting in a long wait time.
- A task could have lower acceptance probability.

These situations could result in one user being markedly worse off in the HOR approach. And if not, they could worsen the practical drawbacks.

A user's ORpair is not expected to be held more than 1-2 times in the mid-p range (see section 6.3.3), but this says nothing about the wait time between rounds. Moreover, when a held ORpair is held again, one of the originally held **offers** or **requests** is satisfied, further decreasing the expected wait time for a user.

Additional drawbacks:

- The acceptance probability for the Hanging ORpair is also likely to change as now two users need to accept whatever match it's in.
- ORpair expiration or tasks that need to be done in good time cannot be handled easily.
- An ORpair whose offer and request need to be done at the same time cannot be held.

7.5.2 *Practical Drawbacks*

Ideally, a sure will upload an ORpair to the offer network, get a match notification, evaluate the option. If the user accepts the match, the user will work on vis offer and wait for vis request. If the user rejects, the user simply waits for another match.

HOR complicates the user experience to facilitate better global performance, and as noted in 7.5.1, HOR risks elongating a user's wait time.

In Figure 14,

- User 3 is asked to receive $\mathtt{task}_d$ and keep offering $\mathtt{task}_c$ for an indefinite period.
- User 1 is asked to give $\mathtt{task}_a$ and wait for $\mathtt{task}_b$ for an indefinite period[57].

First, this complications may make HOR an opt-in feature as there are risks the user has to accept. HOR won't work if there aren't enough users opting in. Practically, users may also have to accept each particular case of creating a new hanging ORpair, to, e.g., prevent a high reputaiton and low reputation user from being joined together: this will make the communication

57 Interestingly, if the offer network is used by computer agents performing speculative execution, this problem diminishes as user 1 may compute $\mathtt{task}_a$ prior to knowing whether the match is accepted or not.

protocol and user experience more complicated, as well as increase the time to determine the acceptance of a match.

Next: what incentive is there for user 3 to keep offering $task_c$ instead of just quitting after receiving request $task_d$? In the kidney case, encouraging anyone to donate a kidney is immoral (and illegal), so exchanges *must* be simultaneous, and chains are started by a donation, so reneging is permissible. In general, a contract is needed: by accepting $task_d$ user 3 accepts to do $task_c$ or else be penalised.

A standard way to do this is via a reputation system[58]. In a reputation system users are given a reputation score, usually based on user feedback. For example, on eBay, feedback describes whether the seller shipped on time and provided the item as described. The reputation score is calculated from the total feedback of the sellar. Users strongly avoid buying from a seller with negative feedback [8]. Sellers with high reputation on eBay can sell the same item at a higher price, or on Taobao[59], which has a thicker market, sell more of an item [41].

In an offer network, users could get feedback on tasks they offer after accepting a match. A user accepting a match and then not delivering will result in negative feedback. Users will be very likely to reject a match with a user with negative feedback[60]. In this way, the reputation system largely prevents users from leaving after they receive their request.

7·5·3 *Offer Reneging and Quees*

The above discussion is about an algorithm users can opt in to *prior* to binding acceptance of a match. Users well, however, accept a match and then renege and not deliver the offered task. A reputation system can make it hard for the user to find new matches in the future, but the user promised vis request needs to be dealt with as well. A hanging ORpair cannot be created with just one hanging request.

Without gifts or asynchronicity[61], the only option is to, unfortunately, ask the user to re-upload his ORpair with a higher weight next time the matching system is run. Or, perhaps, to immediately This would result in the user giving an offer twice; otherwise, the same reneging problem would propogate around the cycle.

A queue, like the cadaver queue in kidney exchange, provides additional options. The user can opt to place vis offer in a reasonable position in the queue. One way for the request queue to be satisfied is via (altruistic) users givig the task for free (as kidney donors do). Another is to allow users to upload asynchronous ORpairs: that is, a user is willing to do the offer for a

58 https://en.wikipedia.org/wiki/Reputation_system

59 A Chinese site like eBay.

60 Reputation can also be part of the matching system; although this makes the entry barrier harder for new users.

61 Asynchronous exchange is mentioned by Abbassi [32], but only implemented via a credit system.

user in the request queue in exchange for the user's place[62]. A chain starting with a gift and ending in the request queue is then also an option. A user could also be willing to accept vis request as a gift prior to vis offer being matched.

7.5.4 *HOR and Asynchronous Exchange*

An alternative to creating a new hanging ORpair is to add the request to the queue and treat the offer as a gift. From this perspective, HOR is, in essence, looking for chains from a specific giver to a specific requester.

A method of finding chains that avoids special processing is to create a dummy task that all lone requests offer and all lone offers request. This would allow for a generalization of HOR where any held offers or requests can be matched up together. This method, however, does not work well with altruistic gifts.

8 EXPERIMENTS

Experiments are run on the author's personal laptop, a Dell XPS3 running Ubuntu 17.04 (Zesty) with 8GB of ram, a 256GB solid state drive, and a 7th generation Intel Core i5 processor (with 4 cores at 2.5GHz).

8.1 Overview of Experiments

The experiments in this thesis aim to explore the performance of optimization algorithms and heuristics on realistic enough offer network models. The run-times of MAX-WEIGHT and GSC-POD pose limitaitons on the size of the experiments that can be done. See Section 8.4 below.

Three randomly generated graphs are used: RB1, BM1, and EN. They are described in the following section.

Performance is measured based on the following metrics:

- (TM): Total number of ORpairs satisfactorily matched.
- (WT): Average wait time of ORpairs that are matched[63].
- (AMS): Average number of matches accepted / suggested matches.
- (SMS): Average size of suggested matches.
- (PoD): Matched task popularity measured by average product of degrees.

62 This is similar to the *you request my house—I get your turn* variant of Top Trading Cycles proposed by Abdulkadiroğlu, Atila and Sönmez [42]

63 Wait time is measured by the number of matching periods an ORpair has been through multiplied by the frequency of matching periods, step size n_{match}.

- (TH): Total number of ORpairs held.
- (HT): Average number of times a hanging ORpair is held.
- (HWT): Max wait time of a user in a held node.

TM, WT, and SMS are standard measures [7] [29] [23]. AMS relates to Section 6.3 on cycle size and acceptance probability: how many users were actually satisfied per user in a suggested match? The lower PoD is, the more hard-to-match ORpairs have been matched.

The first experiments explore the effect of step size, $nMatch$, and the starting size of the graph, $Ninitial$, on performance. GSC and DYN are run up to 35,000 ORpairs to see if the graph size stabilizes. Anderson [29] find their Erdos-Renyi model reaches a steady state at around 2,000 ORpairs and then the size does not increase much. Is there a graph size at which the performance metrics change? Which performance metrics depend on step size? These questions are important for understanding the offer network problem, as well as for choosing the scope of the following experiments.

Taking the first experiments into account, two batches of grid experiments are run on RB1: one up to $Nend = 3,400$ to test MAX-WEIGHT, MAX-2, DYN, GSC, and GSC-POD; and another up to $Nend = 15,000$ to compare DYN, GSC, GSC-POD, and MAX-2. The grid covers a range of acceptance probabilities, HOR or not, and a smaller range of step sizes, $nMatch$.

Due to limited time, only 6 batches sampled from the grid experiments are run on EN and BM1 to check whether the results are graph specific.

Last the relation between acceptance probability and TM is examined.

Note there are more plots than needed to analyse the results. These are included in the Supplementary Materials Appendix 14.

8.2 Graph Parameters

This section describes some graph parameters used in the below experiments.

Three randomly generated graphs are used. Graph (RB1) has task degree distribution similar to Ratebeer. The average difference in in and out degrees of tasks is 1.45, meaning that while the distributions are different, in general popular tasks are popular to both offer and request. See the graph in Figure ??[64] Graph (BM1) has a difference in power-law exponents similar to that of Bookmooch. The average difference is 1.42. Graph (EN) with almost equal power-law exponents for offers and requests. The average difference is 1.36.

RB1 parameters

64 The graphs plotted in this section have been grown to 10,000 ORpairs.

α	γ	β	p_d	δ_{in}	δ_{out}
0.35	0.15	0.5	0.05	1.0	0.5

BM1 parameters

α	γ	β	p_d	δ_{in}	δ_{out}
0.52	0.12	0.35	0.05	0.25	0.05

EN parameters

α	γ	β	p_d	δ_{in}	δ_{out}
0.52	0.12	0.35	0.05	0.25	0.05

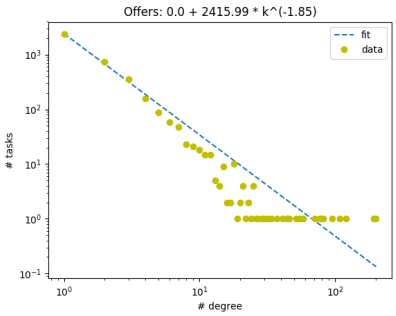


Figure 15: RB1 task offer distribution

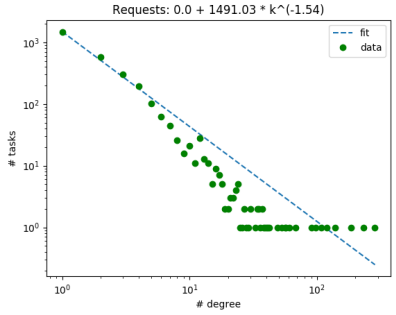


Figure 16: RB1 task request distribution

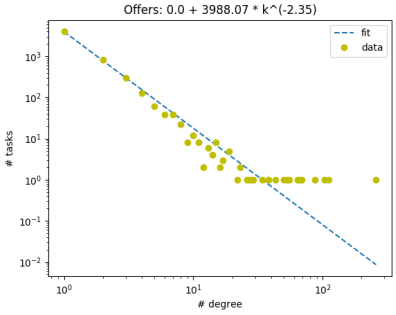


Figure 17: BM1 task offer distribution

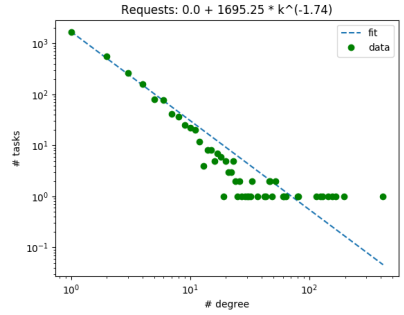


Figure 18: BM1 task request distribution

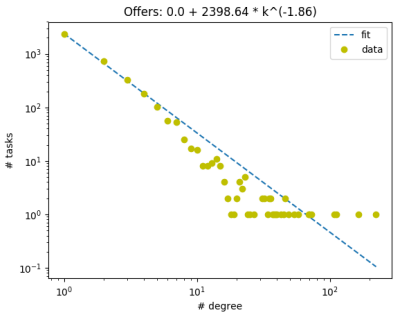


Figure 19: EN task offer distribution

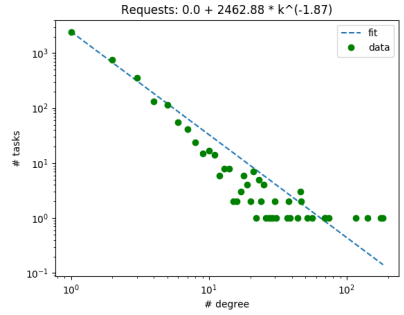


Figure 20: EN task request distribution

8.3 Graph Generation

For each experiment a series of graph updates is computed using the model in section 6.1.2. This series is is used for each test in a single experimental run to isolate variance in the run. However, a new series is generated with the same parameters in different experiments to minimize dependence of results on any one randomly generated graph.

The matching algorithm is run after $N_{initial}$ ORpairs are added and their statistics are not preserved. Wait time counts are set to -1. This trade-off allows the performance at any given point in the offer network's evolution to be better tested. See Section 8.7.

8.4 Run Time

Each of the 4 tests below are run on different graph update series.

DYN, TWO, and GSC are run on RB1, BM1, and EN with $N_{initial} = 100$, $N_{end} = 15,000$, $n_{match} = 100$, and $p = 1$ to measure match time as a function of the number of ORpairs in the graph. See Figure 21. GSC and GSC-POD are then run with the same settings in Figure 22. DYN and CSC are also run with $n_{match} = \{1, 20\}$ in Figure 23. The time test including MAX-WEIGHT, being slower, is only run up to $N_{end} = 5000$: Figure 24.

DYN, TWO, GSC, and GSC-POD all run in linear time with respect to the graph size, and MAX-WEIGHT runs in $\mathcal{O}(N^3)$ time. In practice, the time to find shortest cycles seems to depend on step size [65].

GSC is comparable to TWO. DYN scales very well with graph size compared to GSC, as performance largely depends on the number of newly adde nodes. GSC-POD scales worse, reaching 60 seconds while GSC runs in 1.5.

65 Note this could be tested more rigorously.

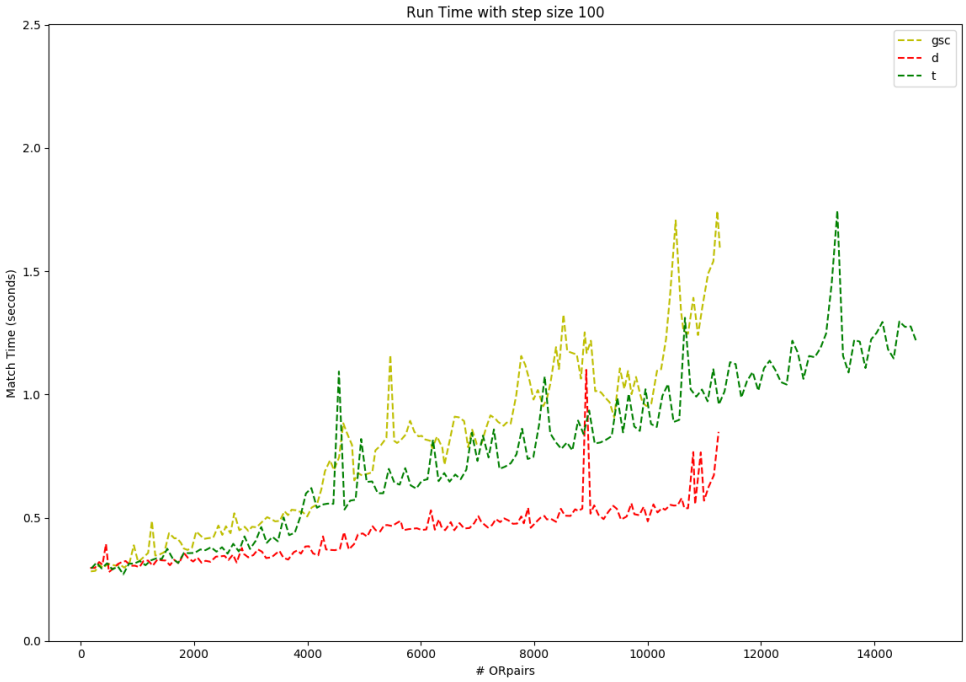


Figure 21: EN - Run Time - Step Size 100

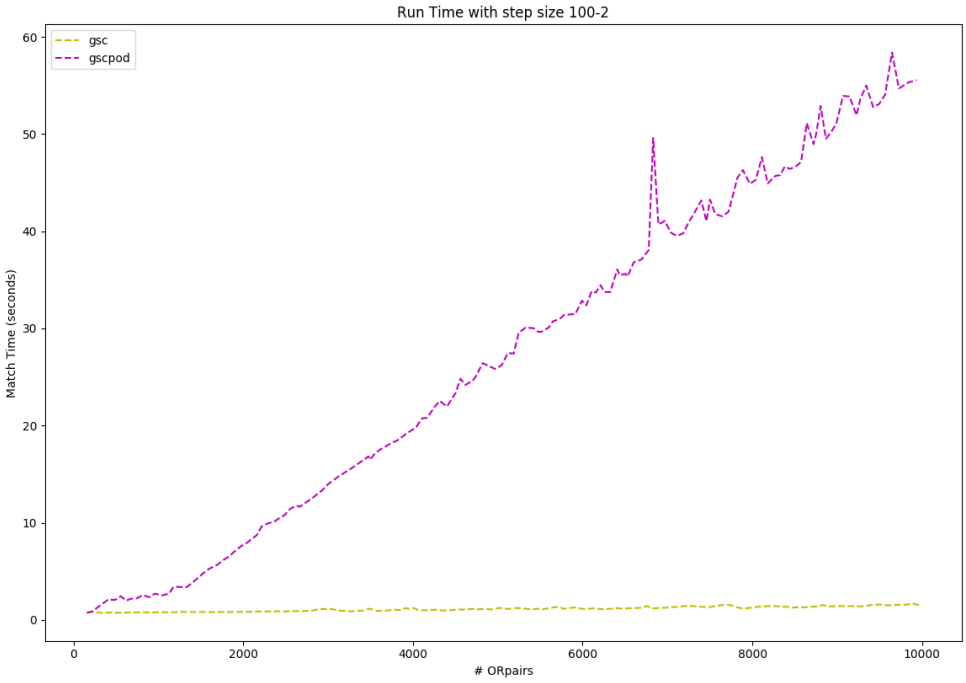


Figure 22: RB1 - Run Time - GSC-POD

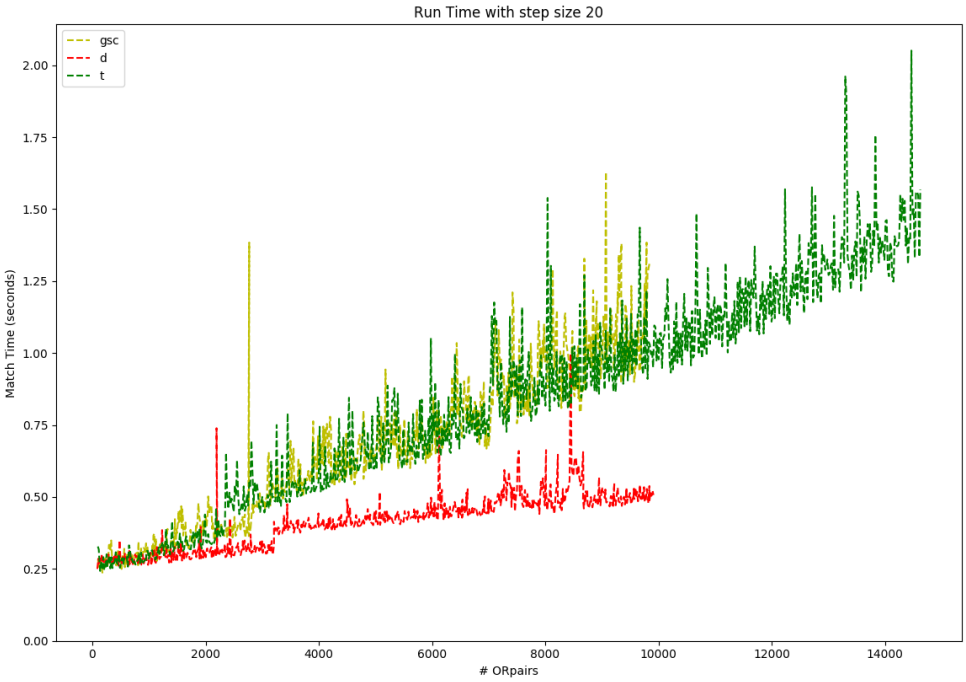


Figure 23: RB1 - Run Time - Step Size 20

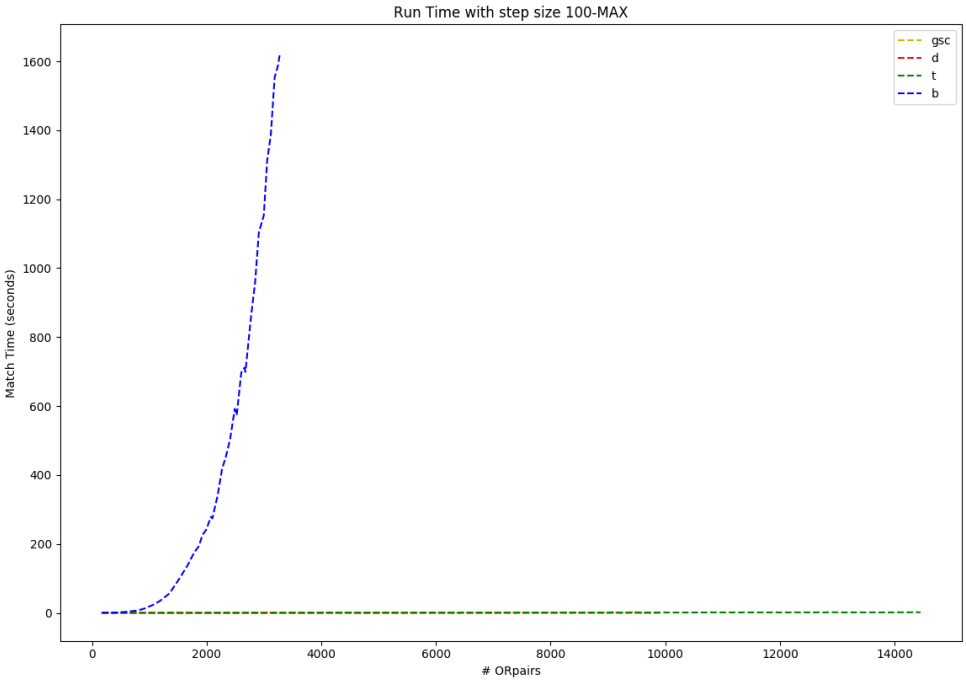


Figure 24: RB1 - Run Time - MAX-WEIGHT

8.5 Effect of step size, n_{match}, on performance

The matching algorithms (GSC, DYN, MAX-WEIGHT, MAX-2) are run on $n_{match} \in \{5, 10, 20, 40, 80, 120, 240, 480, 960\}$ with $N_{initial} = 100$, $N_{end} =$

3003, and $p = 1$. Initially, I generated a graph series for each matching algorithm run, which can be seen in Figures 25 and 26.

Note that in the first run TM (toal matched) does not depend much on match frequency MAX-WEIGHT, which can have very high match sizes has a small average MS comparable to the short cycle seeking algorithms. MAX-2 performs the same regardless of the match size, implying there are limtied 2-way swaps.

GSC and DYN performance decreases slightly with match frequency. MAX-WEIGHT's performance increases, but this comes at the cost of more than doubling the WT (wait time).

Another run where all matching algorithms are tested on the same generated graph series is done with $p = 1$, $N_{initial} = 100$, $N_{end} = 5000$, and $n_{match} \in \{1, 2, 3, 4, 5, 10, 20, 40, 60, 80, 100, 125, 250, 500, 1000\}$. See Figures 27 and 28. The variance in performance diminishes.

The experiments show that the particular step size does not matter much for performance, provided $n_{match} \leqslant 50$, moreover MAX-WEIGHT may perform comparably to GSC and DYN with low acceptance probability p

Unfortunately[66], the way WT is measured, an ORpair added i steps before matching will be assumed to have waited for roughly $n_{match} - i$ steps longer than it has. This will lead to an average error of roughly a $n_{match}/2$ longer average wait time. The effect will be more pronounced when $(N_{end} - N_{initial})/n_{match}$ is small. This correction seems to bring the WT for high step sizes to par, or only a couple hundred steps, higher than for low step sizes. This implies that, contrary to expectations and the Erdos-Renyi model in Anderson [29], small step sizes ($n_{match} \leqslant 50$) do not out-perform large ones much.

Interestingly, $n_{match} = 10, 20$ can outperform $n_{match} = 5$ on both WT and TM metrics.

66 As the author does not have time to add a more nuanced wait time measurement.

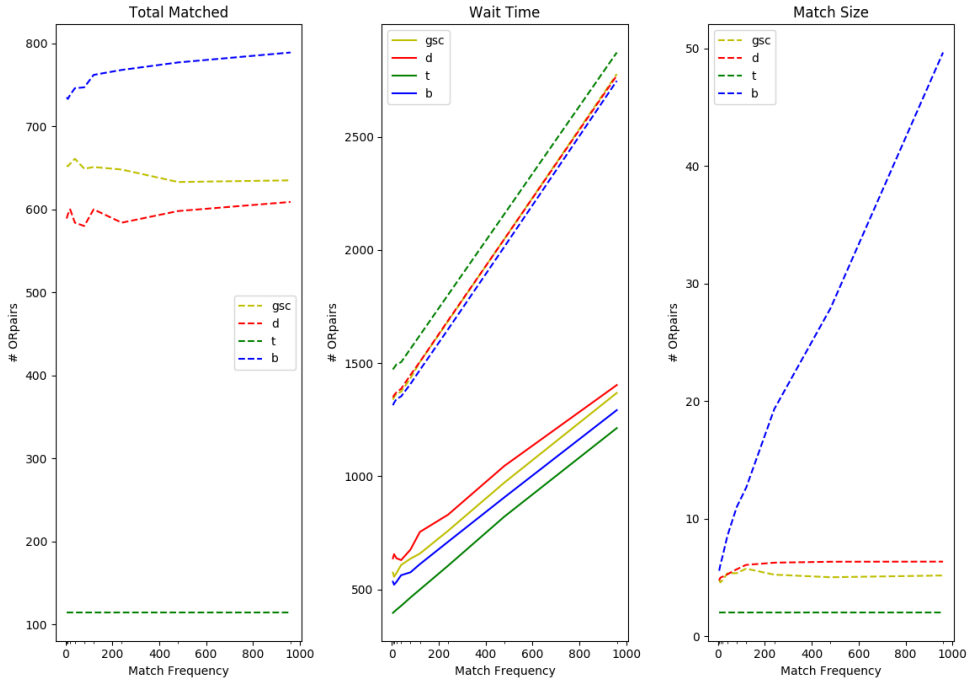


Figure 25: BM1 - Step Size Test 1
The dashed lines are the WT of unmatched ORpairs, and the solid lines matched ORpairs.

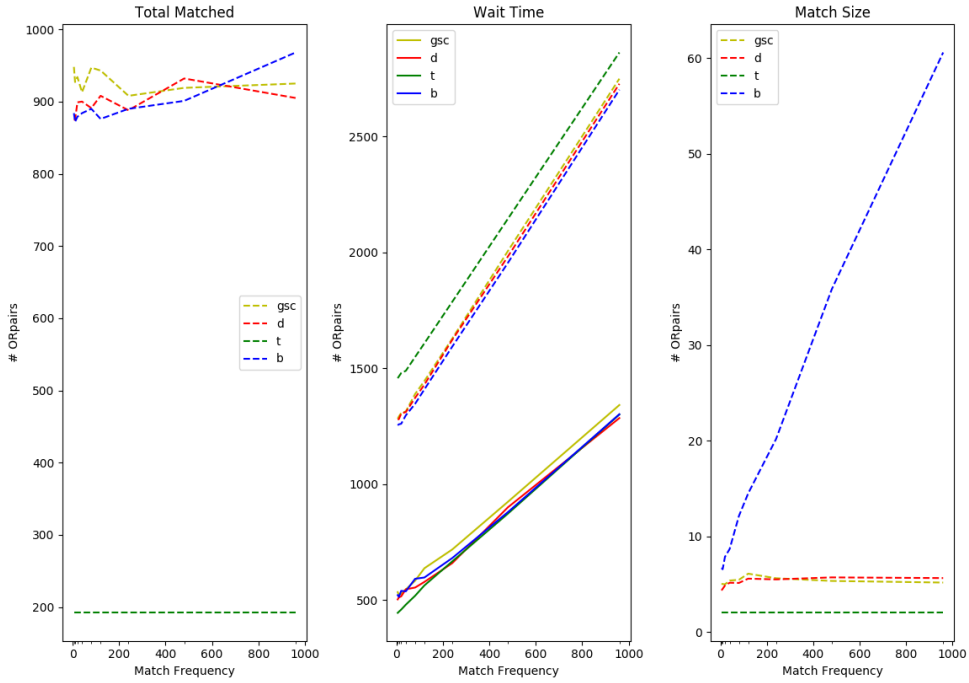


Figure 26: RB1 - Step Size Test 1

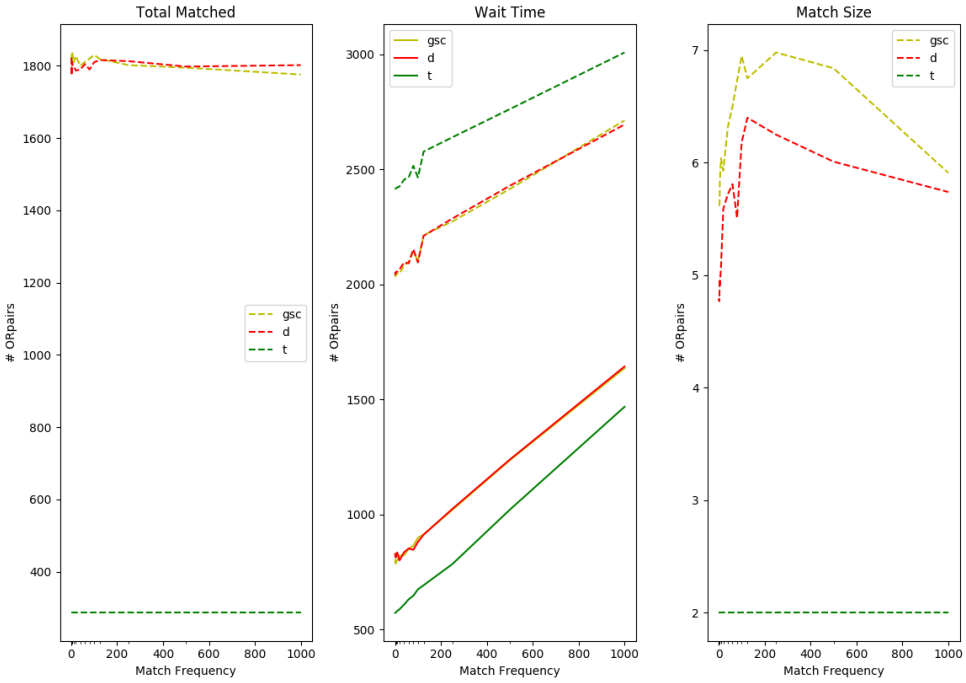


Figure 27: RB1 - Step Size Test 2

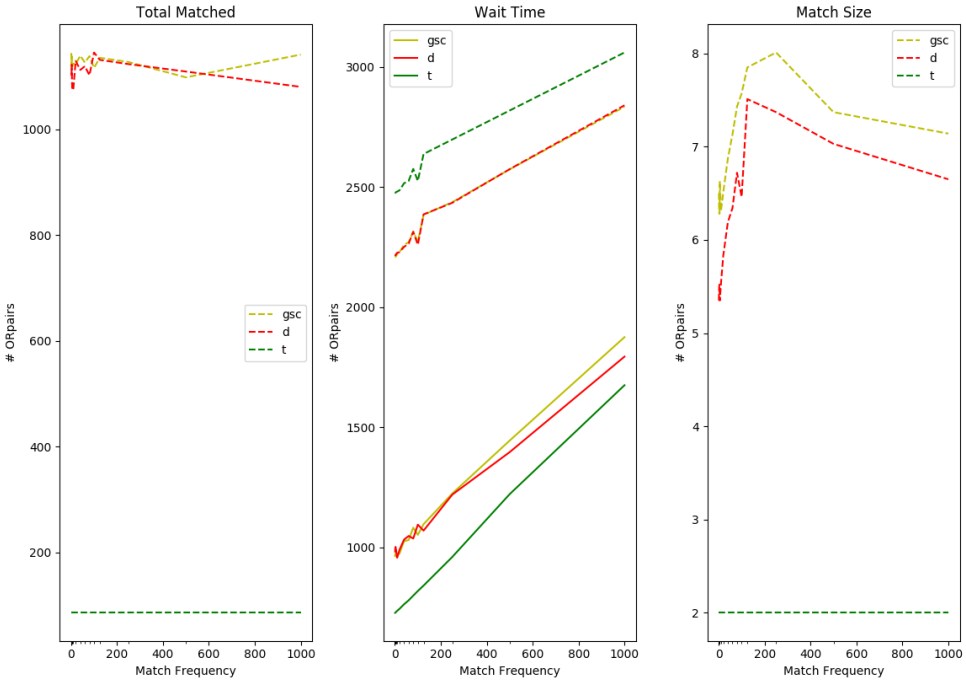


Figure 28: EN - Step Size Test 2

8.6 Graph Size Over Time

DYN is run on RB1, BM1, and EN from 100 to 35,000 ORpairs with $p = 1$ and $nMatch = 20$. GSC is also run on EN. The graph size grows linearly in added ORpairs, with a coefficient less than one.

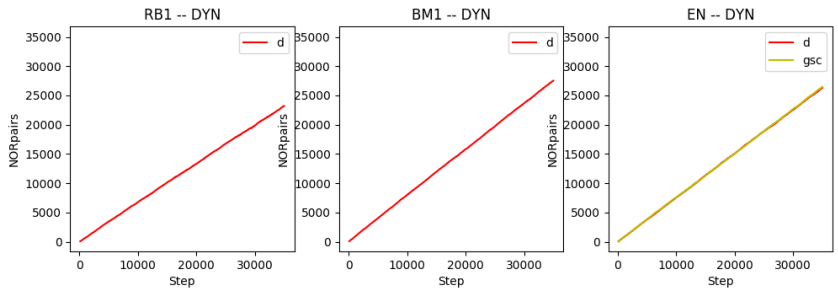


Figure 29: Graph Size vs ORpairs Added

8.7 Effect of initial graph size, $N_{initial}$, on performance

Matching algorithms (TWO, DYN, GSC) are run for 3000 steps starting at $N_{initial} \in \{10, 50, 100, 200, 400, 600, 800, 1000, 1200, 1600, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000, 11000\}$ with step size $n_{match} = 20$ and $p = 1$. [67]

The TM (total matched) ORpairs is near constant for DYN and GSC, and decresase for TWO. MS (match size) increases, more for GSC, with graph size. Matched and unmatched WT (wait time) rapidly increases and appears to level off to around double the low $N_{initial}$ WT. Thus a stable state with respect to throughput[68] seems to be reached for $N_{initial} > 10,000$. As in Section 8.6 the graph size does not stabilize, this is likely a balance of two factors:

- There are more ORpairs to be matched with as graph size increases.
- There are fewer cycles containing any two non-popular nodes as graph size increases.

First, it's worth doing experiments with large enough graphs that wait time has leveled off.

However, this is not feasible with MAX-WEIGHT. Fortunately, the dependence on $N_{initial}$ appears uniform among matching algorithms.

67 The graph sizes are too big for MAX-WEIGHT, but the trends likely to be similar. It's possible, however, MAX-WEIGHT will take better advantage of larger graph size.

68 Through put is the number of matched ORpairs in a given timeframe, that is, TM and WT

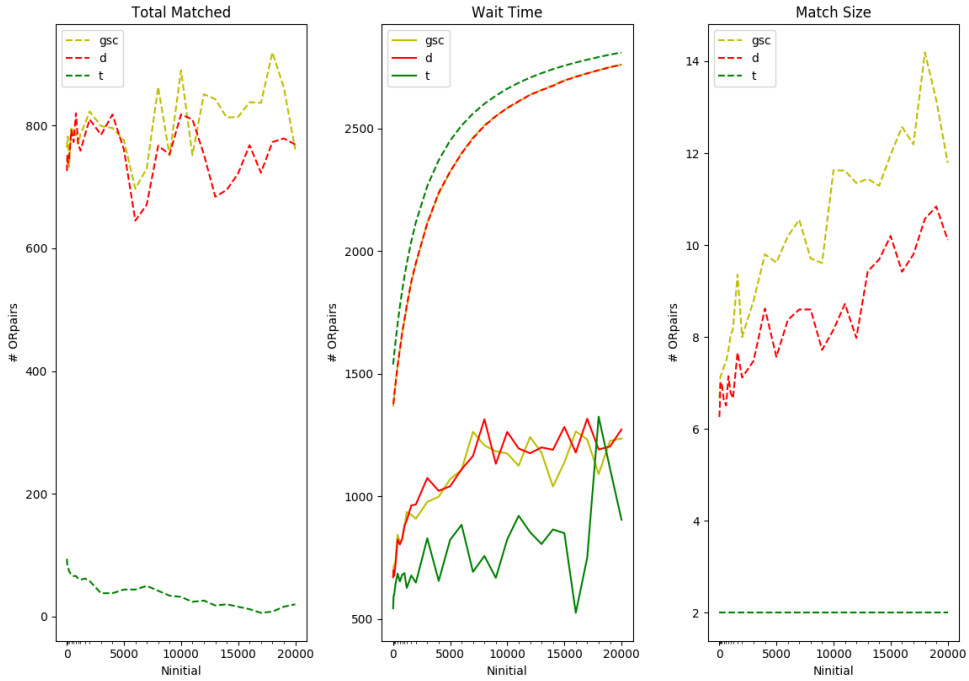


Figure 30: Ninitial Test - EN

8.8 Performance Test – Low $N_{initial}$

This experiment is run from $N_{initial} = 400$ to $N_{end} = 3400$. Without the HOR approach with held ORpairs only $p = 0.9$ is tested. With HOR, $p \in \{0.3, 0.5, 0.7, 0.8, 0.9\}$ are tested.

The matching algorithms are run with the following step size values:

Matching Algorithm	n_{match}
GSC	{1, 5, 10, 25, 50, 100, 250}
DYN	{1, 5, 10, 25, 50, 100, 250}
GSC-POD	{5, 10, 25, 50, 100, 250}
MAX-2	{5, 10, 25, 50, 100, 250}
MAX-WEIGHT	{40, 100, 250, 500}

For the plots in this and the following section, remember that solid lines for wait time, task popularity (measured by PoD, product of degrees), and hold times. Total matched (solid) and total held (dashed) ORpairs are shown on the same plot. SMS (average suggested match size – solid) and AMS (average accepted matche size / SMS – dashed) in the Match Size plot.

First, see Figure 31 for the case without HOR. MAX-WEIGHT performs significantly worse than the other algorithms o all metrics. As indicated in MS, AMS is low. Recall that the expected number of satisfied users is: $\mathbf{E}[c_k] = k(0.9)^k$, and in Section 8.5 MAX-WEIGHT's MS increases rapidly with match frequency. Thus even though SMS is not much higher than for GSC at low n_{match}, each rejected cycle will engender the same effect

as having a larger n_{match}, a vicious cycle [69]. No further experimentation is needed for MAX-WEIGHT without HOR. The other algorithms will be tested without HOR on larger graphs.

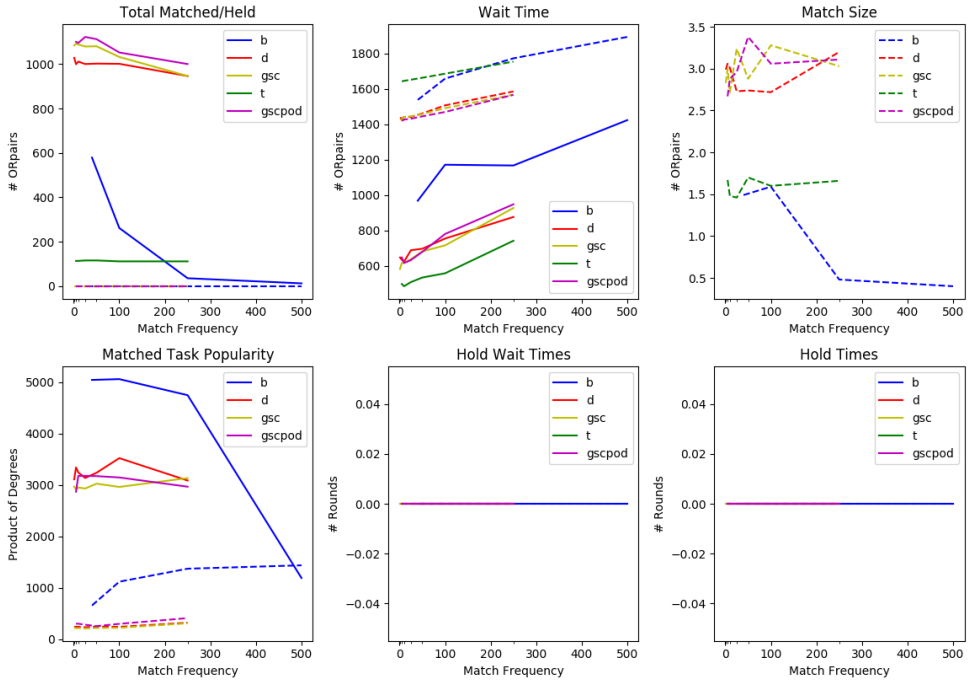


Figure 31: Performance Test - RB1 - $p = 0.9$ without HOR

The trend in MAX-WEIGHT's performance as p decreases can be seen in Figures 32 - 36

69 It's possible for $n_{match} \leqslant 10$, MAX-WEIGHT will perform well. The algorithm is, however, too slow for matching every new ORpair to be considered seriously.

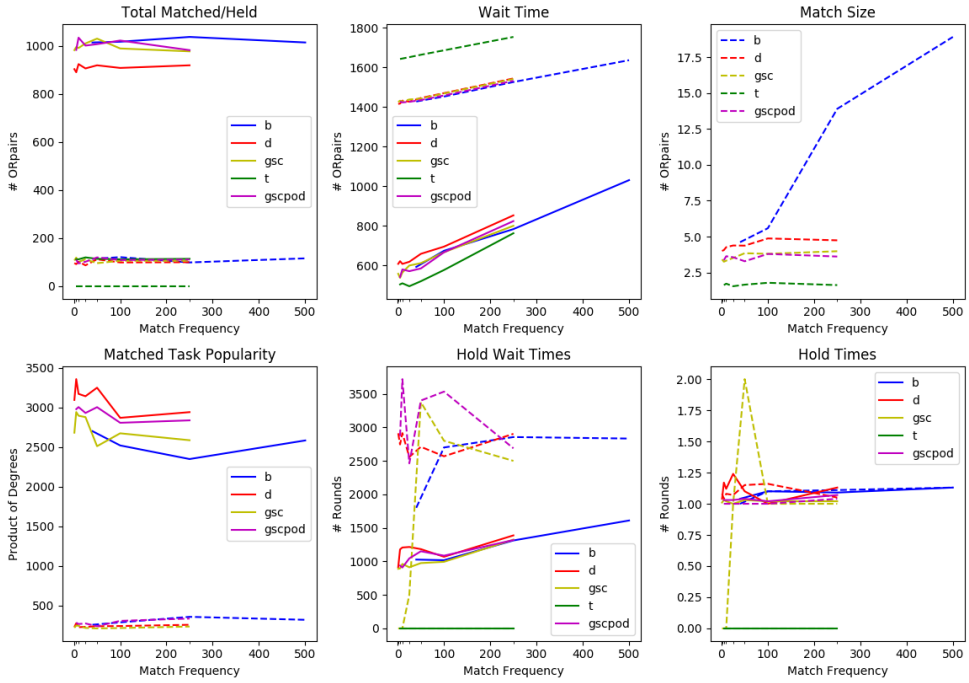


Figure 32: Performance Test - RB1 - $p = 0.9$ with HOR

At $p = 0.9$, MAX-WEIGHT and HOR outperforms the other algorithms in TM, WT, and PoD. HWT (hold wait time) is only about 1.5 times the average wait time, and nodes are held more than once are rare. Larger match sizes seem okay.

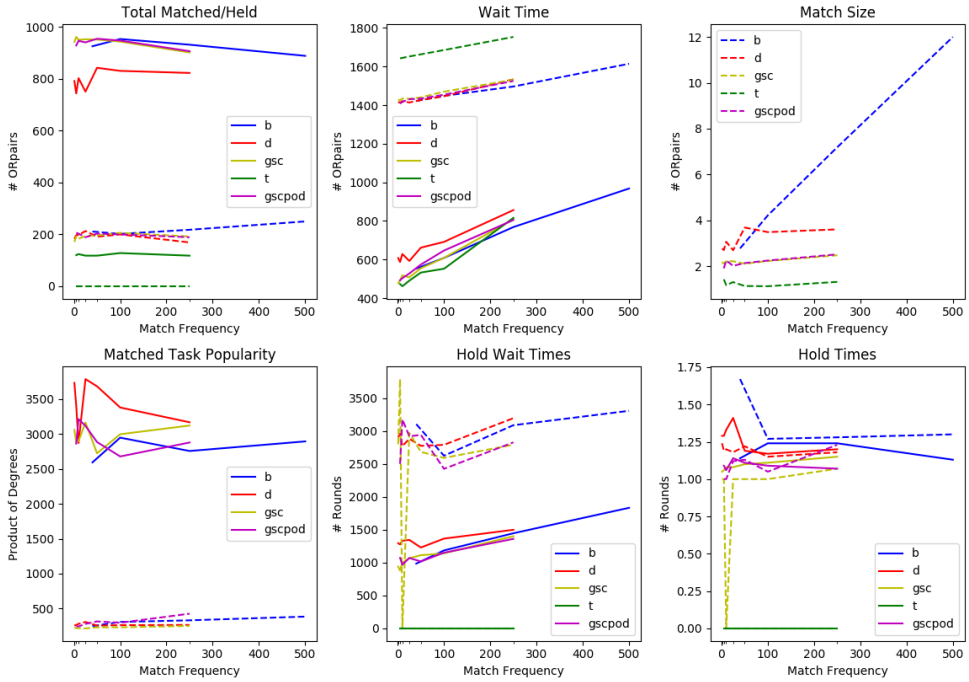


Figure 33: Performance Test - RB1 - $p = 0.8$ with HOR

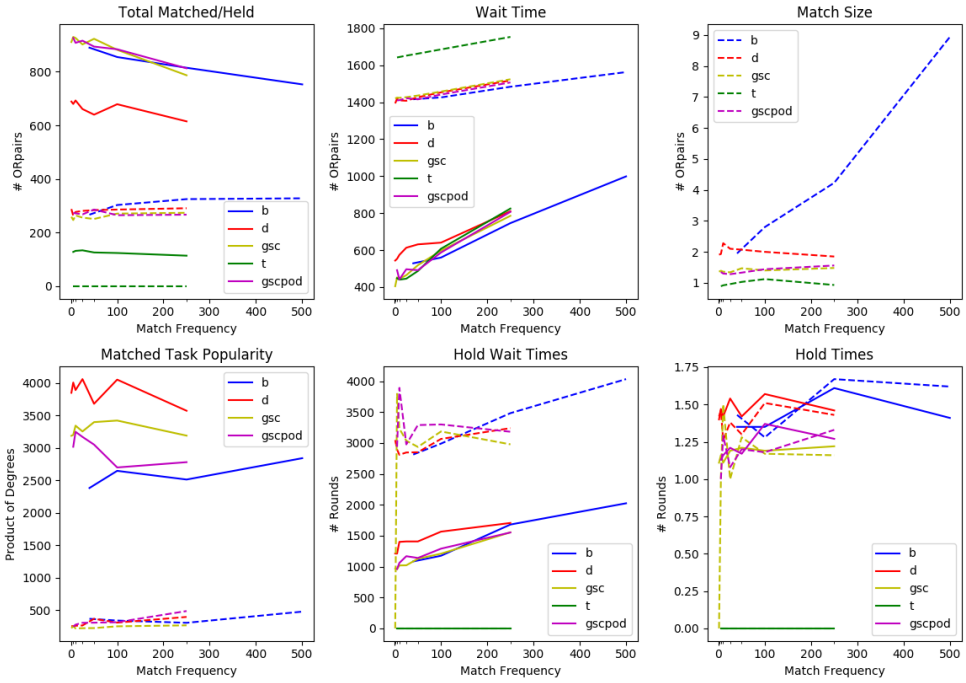


Figure 34: Performance Test - RB1 - p = 0.7 with HOR

At p = 0.8 and p = 0.7, MAX-WEIGHT performs on par with generally better PoD. AMS decreases, and more nodes are held. HWT about 2 times WT.

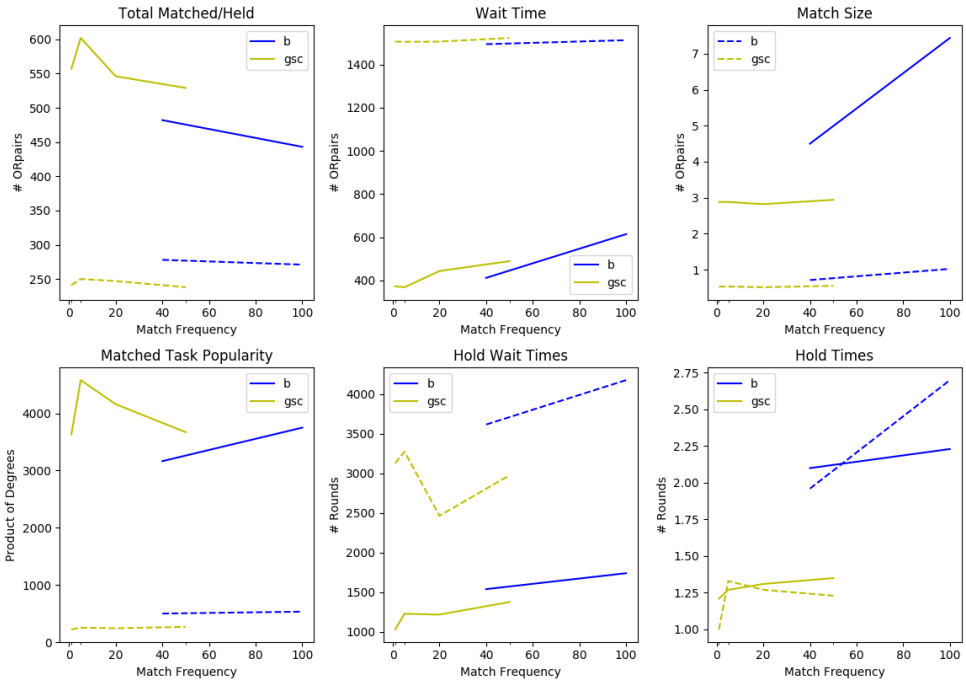


Figure 35: Performance Test - BM1 - p = 0.5 with HOR

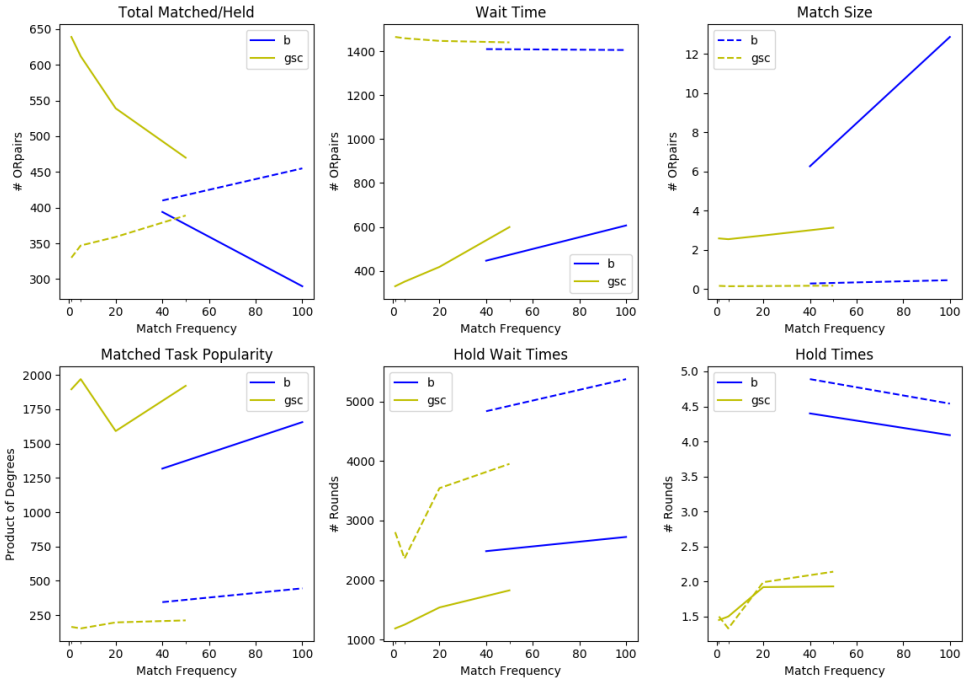


Figure 36: Performance Test - EN - $p = 0.3$ with HOR

At $p = 0.5$ and $p = 0.3$, GSC out-performs MAX-WEIGHT: short cycles become important in spite of HOR. By $p = 0.3$, held nodes are held for an average of 4.5 rounds. Interestingly, despite performing worse, MAX-WEIGHT achieves better PoD. This may be due to the *optimality* of the suggestsed matching. Note also that AMS gets very low, near zero, even for GSC.

Another trend is that small step sizes performs increasingly well as p decreases: with high cycle rejections and held ORpairs, there are more often potential matches. Thus suggesting matches with high frequency works well.

This experiment batch shows that HOR can be an effective approach and workaround to mid and high $p \geqslant .07$. MAX-WEIGHT, which performs dismally without HOR, performs better than the other matching algorithms with HOR. Even in low p ranges, MAX-WEIGHT only matches a couple hundred fewer ORpairs than GSC and has comparable wait time.

8.9 Performance Test – High $N_{initial}$

This experiment, without MAX-WEIGHT, is run from $N_{initial} = 10,000$ to $N_{end} = 15,000$. The bulk of the experiment is done on RB1, with 2 on EN and 2 on BM1 to verify that the results do not depend on the specifics of RB1. Without holding ORpairs, $p \in \{0.3, 0.5, 0.7, 0.9\}$; and with HOR, $p \in \{0.3, 0.5, 0.6, 0.7, 0.8, 0.9\}$.

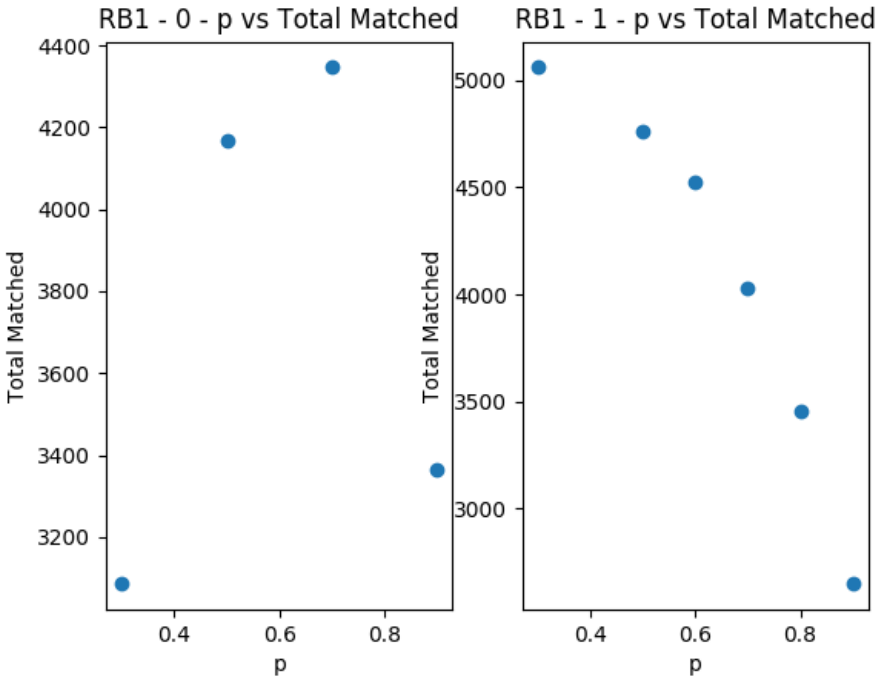
The matching algorithms are run with the following step size values:

Matching Algorithm	n_{match}
GSC	{1, 5, 25, 50, 100}
DYN	{1, 5, 25, 50, 100}
GSC-POD	{50, 100}
MAX-2	{5, 25, 100}

MAX-2 is only run for $p < 0.7$; however its performance is always dismal.

8.9.1 *Acceptance Probability and Performance*

The following plot is made with the best performing matching algorithm's best n_{match} setting where 0 is without HOR and 1 is with HOR.



First, note that even without HOR, the best performance is achieved with $p = 0.7$ and $p = 0.5$! This result is surprising. Randomly rejecting matches with high probaility (98% chance of rejection for 10-cycles at $p = 0.7$) may be harnessing some of the effect Dickerson [7] noted with vertex/edge/cycle/graph potentials. However, the way these are utilized seems to be by repeatedly trying larger matches until one gets lucky, which is only a feasible strategy is querying users' acceptance is cheap.

With HOR, TM and WT improve as p decreases, as does GSC-POD's relative performance and PoD. The relative advantage of low step sizes diminishes. A popular task can, via being held or rejected, facilitate multiple exchanges before being part of an accepted ORpair. Curiously, these results imply that for high p randomly rejecting some suggested matches could be a good heuristic, very reminiscent of Dickerson's potentials method.

While HOR's best performance is higher than without, curiously, HOR seems counterproductive at high p values. With HOR when a user rejects a match, 3 users go unsatisfied in that round. It seems that with $p = 0.9$ simply rejecting and finding a new match is better. As seen later in Figure **??**, the average match size for GSC-POD at $n_{match} = 50$ is 10, the optimal for $p = 0.9$ with an expected number of satisfied users of 3[70].

8.9.2 *Without HOR*

First note that trend of small n_{match} performing better as p decreases holds for high $N_{initial}$ too without HOR in Figures 37 - 40

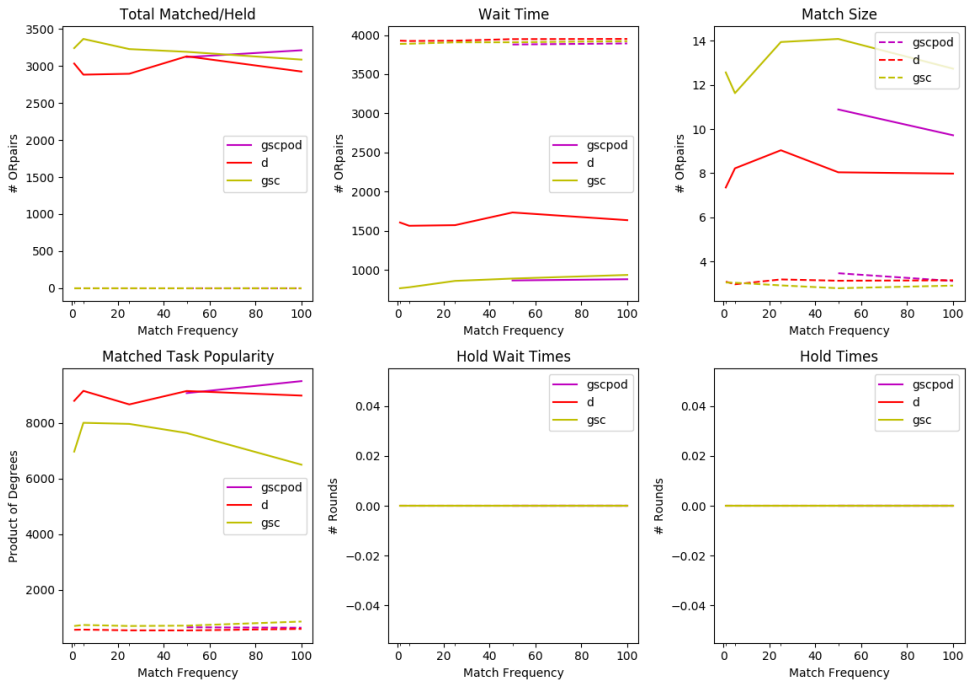


Figure 37: Performance Test - RB1 - $p = 0.9$ without HOR

As in the $p = 1$ case, step size does not matter much and GSC, DYN, and GSC-POD perform about equally. GSC matches more unpopular tasks despite GSC-POD prioritizing them: perhaps in the scale free graph unpopular tasks are more likely to be connected to popular tasks?

70 Dicussed in Section 6.3

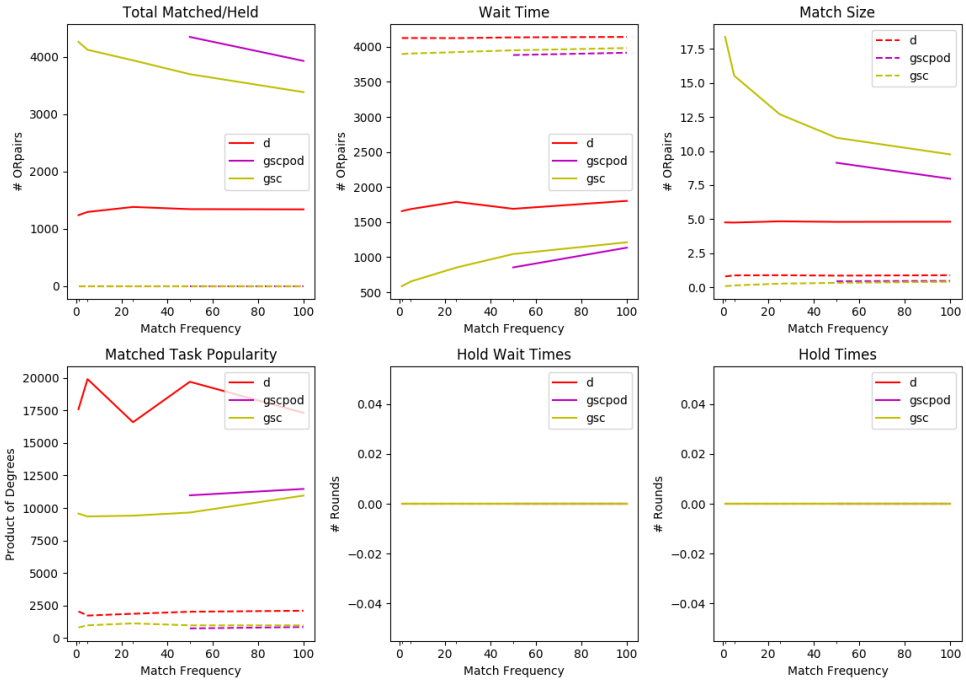


Figure 38: Performance Test - RB1 - $p = 0.7$ without HOR

Step-size starts becoming significant. Oddly, GSC ends up with very high SMS (suggested match sizes): GSC-POD performs better even with $n_{match} = 50$. Both have as good WT as with $p = 0.9$, despite the AMS being near zero instead of 3 (that is, on average a suggested match resulted in 3 satisfied ORpairs). First, this means many more matches are being suggested. The thicker graph then semes to allow ORpairs to get matched as fast despite multiple attempts being needed on average.

DYN starts performing poorly. This is likely because rejected ORpairs are not treated as new nodes by the dynamic matching.

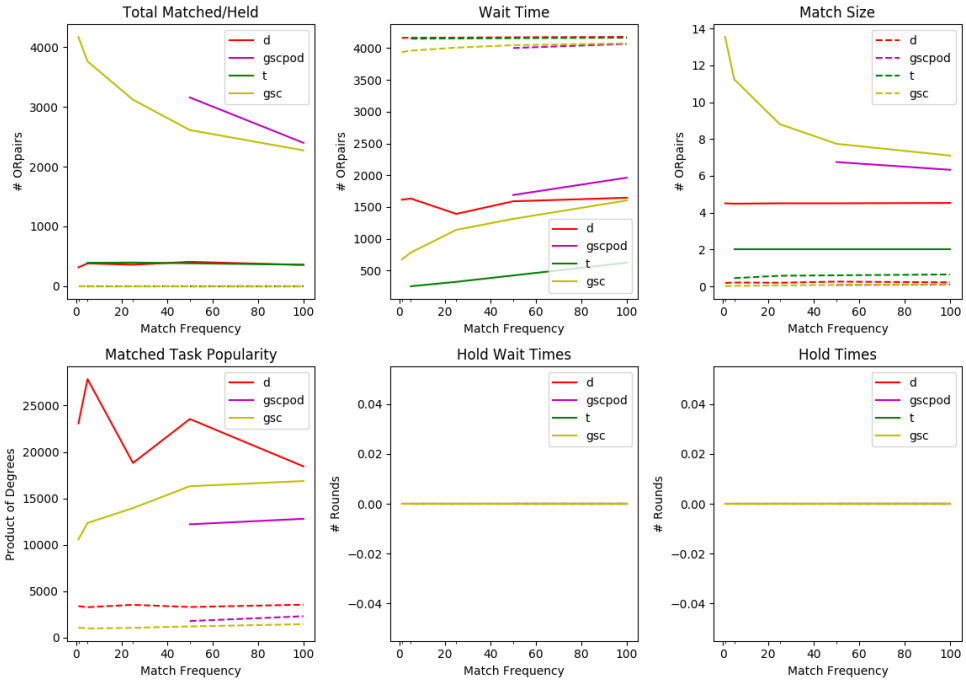


Figure 39: Performance Test - RB1 - p = 0.5 without HOR

Indications are GSC-POD would out-perform GSC if run with smaller step size; however, even at large step size, GSC has better WT.

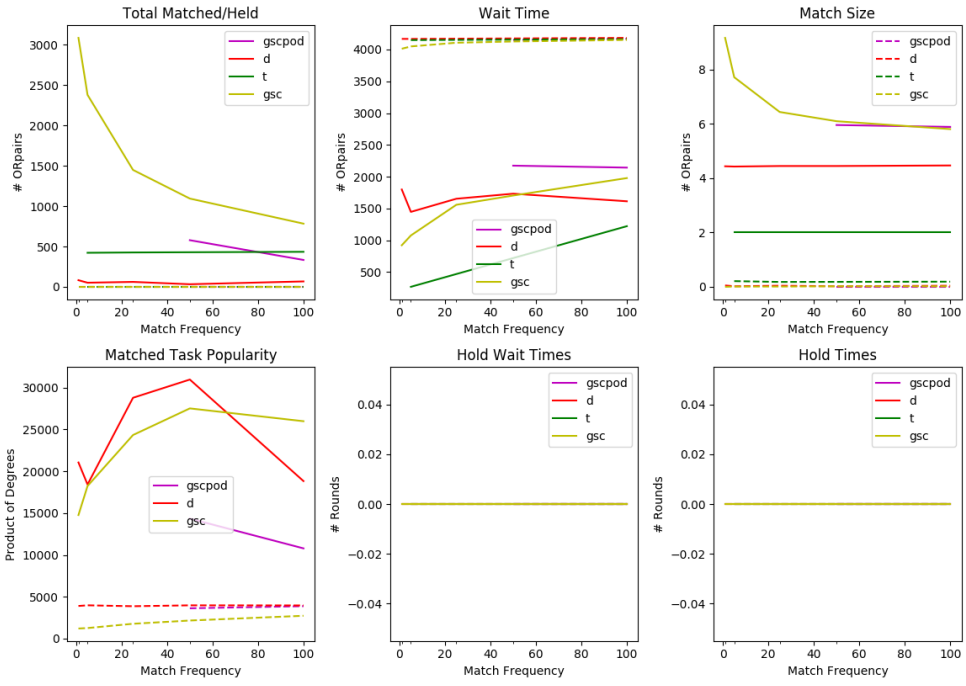


Figure 40: Performance Test - RB1 - p = 0.3 without HOR

GSC performs significantly better than GSC-POD. The WT, however, has doubled or tripled: very many matches are needed to get an accepting match, which doesn't happen unless step size is small.

8.9.3 *With HOR*

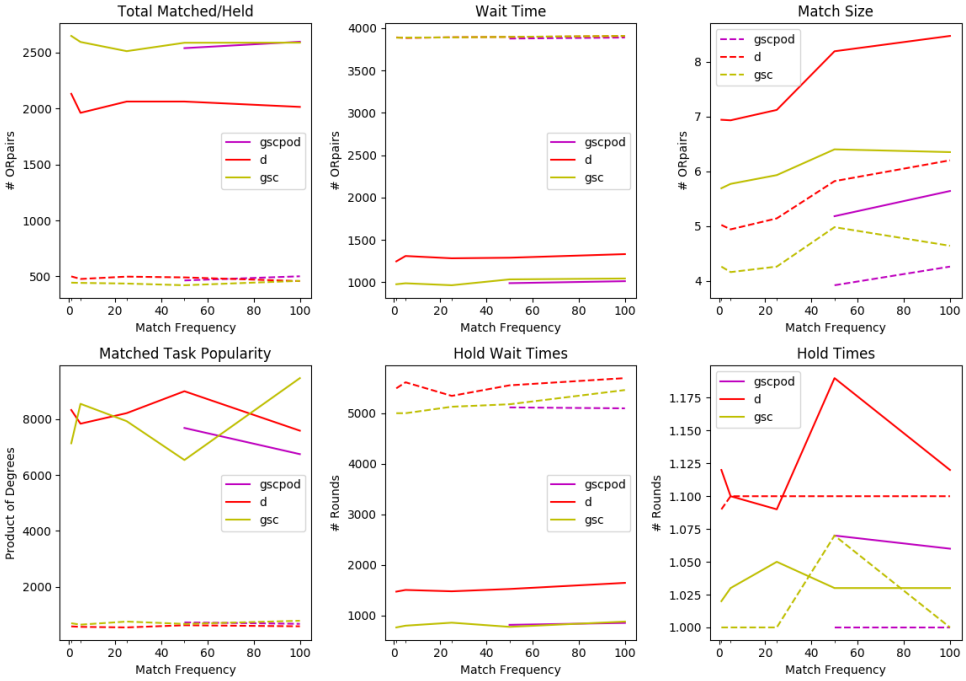


Figure 41: Performance Test - RB1 - p = 0.9 with HOR

As in the p = 1 case, step size does not matter much and GSC, GSC-POD perform about equally.

DYN already performs worse: likely because held nodes are not treated as newly added nodes.

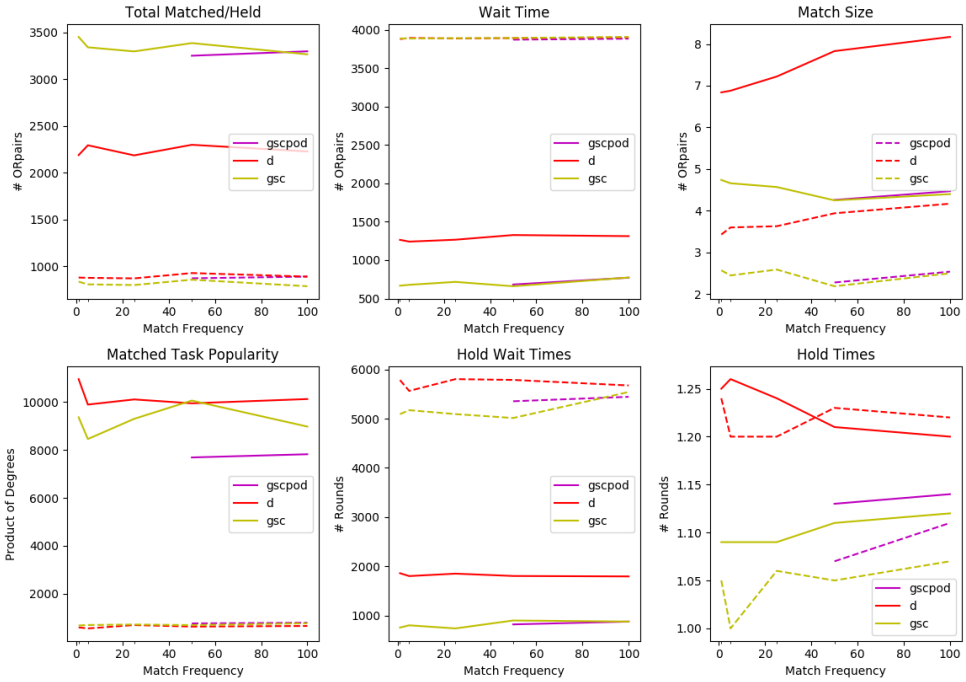


Figure 42: Performance Test - RB1 - p = 0.8 with HOR

Same as with $p = 0.8$, except GSC-POD has slightly better PoD.

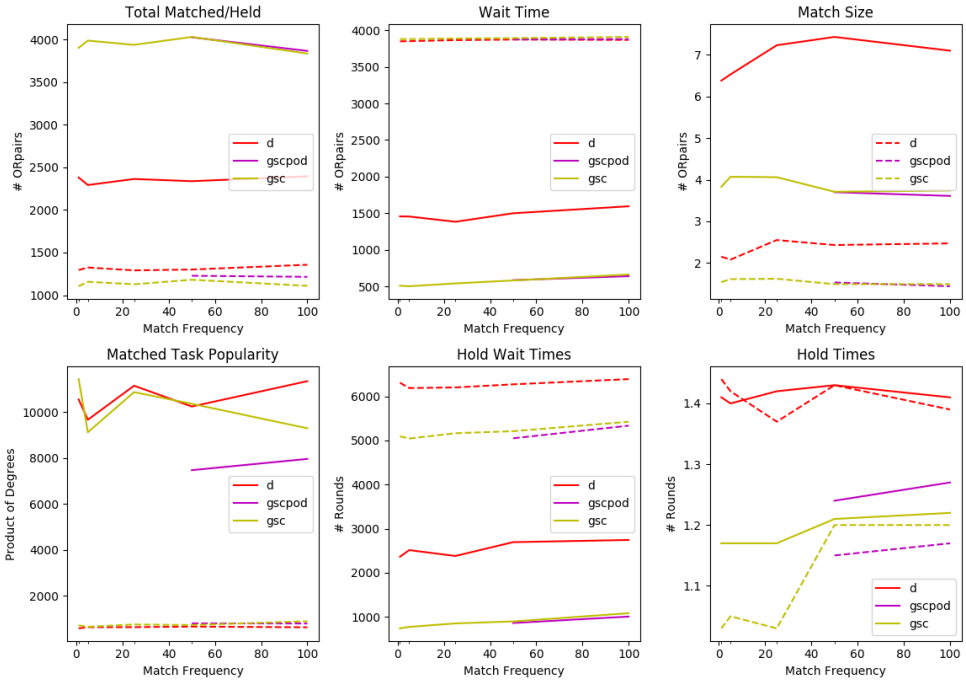


Figure 43: Performance Test - RB1 - $p = 0.7$ with HOR

GSC-POD's PoD advantage increases slightly.

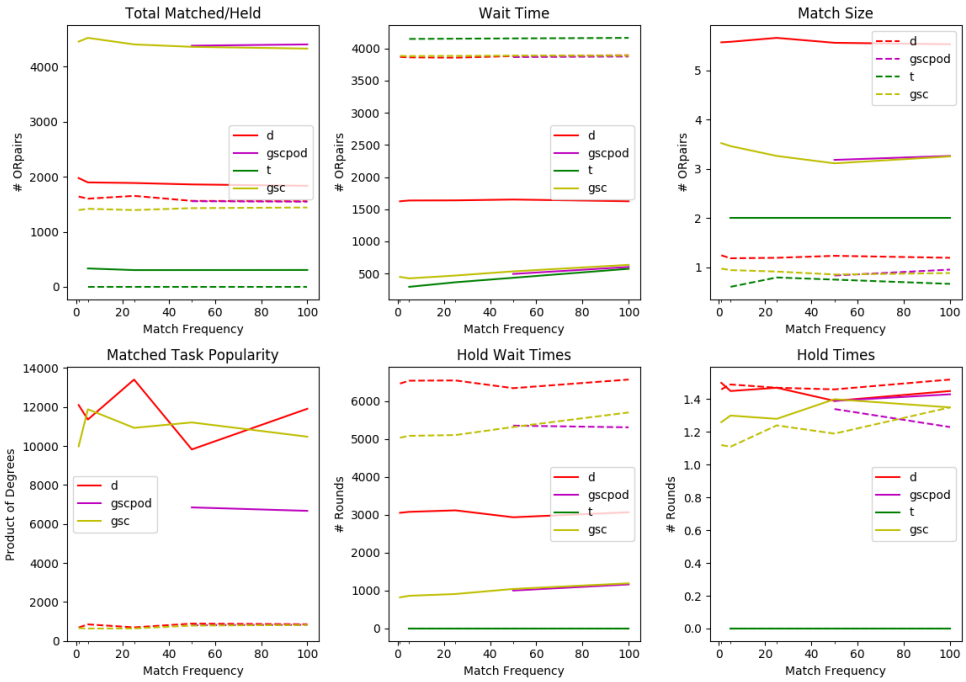


Figure 44: Performance Test - RB1 - $p = 0.6$ with HOR

GSC-POD's PoD performance increases even more.

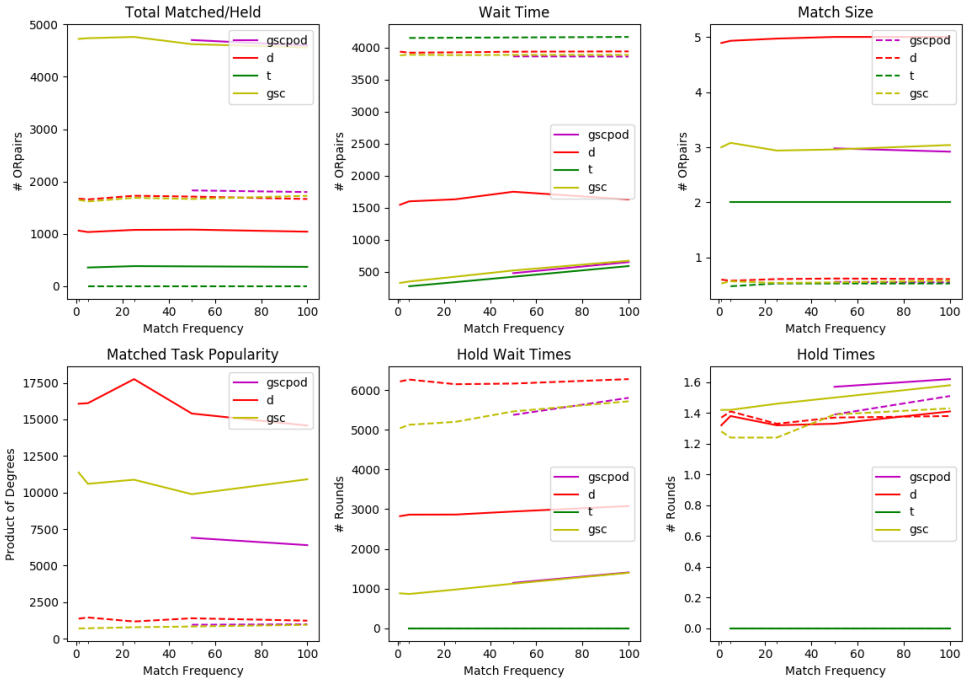


Figure 45: Performance Test - RB1 - $p = 0.5$ with HOR

GSC-POD's TM may be slightly better than GSC's now.

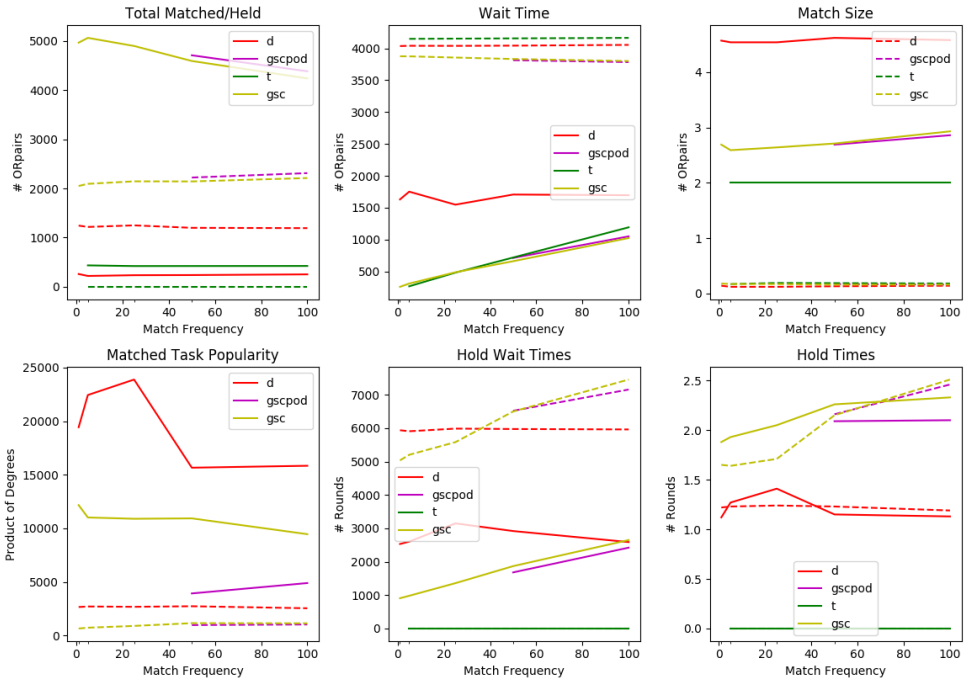


Figure 46: Performance Test - RB1 - $p = 0.3$ with HOR

GSC-POD's TM trajectory slightly higher than GSC's, and PoD performance significantly better. GSC-POD seems to perform best with low p and HOR, perhaps because this leads to hard-to-match tasks being shuffled around more.

Small step size does not confer much performance benefit for $p \geqslant 0.5$. This is promising as algorithms such as GSC-POD that take additional factors into account are slower (even if still linear in the graph size). This can also be because new nodes are created with HOR, and thus the same cycle can't be repeated over and over again [71]

Wait time trends are interesting: WT decreases with p from 0.9 to 0.5 and HWT increases as p decreases. HWT does not increase much over the WT of about 1000 for $p \geqslant 0.9$. This supports the increased dynamism hypothesis behind higher performance for lower p.

9 DISCUSSION OF RESULTS

This thesis aims to explore how feasible offer networks[72] are with state-of-the-art matching algorithms. Exponential time algorithms, as used in kidney exchange networks, are both too specialized and slow to scale.

9.1 Matching Algorithm Feasibility Comparison

Offer networks with only MAX-2 are not feasible. Over a run of 3k or 5k steps (in n_{match} tests), the algorithm matches under 200 ORpairs. Moreover, the number of matches by MAX-2 decreases as the size of the graph increases (in the $N_{initial}$ test). Less than 5% of ORpairs will be suggested a match

MAX-WEIGHT performs almost as poorly as MAX-2 withou $p = 0.9$, and worse for $n_{match} > 200$. The algorithm also runs in $\mathcal{O}(|ORpair|^3)$ time and cannot feasibly be run frequently on a large scale.

For high p, the performance of GSC, GSC-POD, DYN, and MAX-WEIGHT all perform similarly in total matchings, wait time, and hold wait times. GSC-POD does a better job of matching hard-to-match (unpopular) ORpairs with HOR, but not without. MAX-WEIGHT appears to perform even better than GSC-POD for matching hard-to-match ORpairs: maximal cycle-covers include more hard-to-match ORpairs than biased greedy cycle covers. This makes sense as, recalling the distribution, 1/3 of **tasks** have 3 or fewer **offers**: a maximal covering will have to utilize hard-to-mask ORpairs.

71 However, a held ORpair could be matched up in a 2-cycle with the rejecting ORpair that triggered its formation. Both fortunately and interestingly, this does not seem to be happening excessively. An algorithm that abused this lazy implementation feature could suggest the same matches as for $p = 1$, and then *fill in the gaps* in the next few rounds by suggesting this 2-cycle until it's accepted. Yet this alone would lead to the same performance as with $p = 1$, except longer delays in hold wait time (HWT). The average HWT is around 1.5x to 2x the average wait time (WT), indicating held ORpairs are actually rematched. Moreover, total matched (TM) and WT performance are actually *better* with mid-p ranges. So it would appear there is more going on.

72 barter exchange networks

The experiments, contrary to Anderson's findings [29], indicate that performance in total matched, wait time, and matched task popularity (PoD) do not decrease much for $p \geqslant 0.7$. Thus for mid and high p ranges, a performance-optimized implementation of MAX-WEIGHT appears a feasible choice of matching algorithm.

However, the greedy heuristics perform almost as well as MAX-WEIGHT in a fraction of the time, making them better choices for large scale offer networks. DYN runs significantly faster than GSC and tends to have smalle suggested match sizes. An updated DYN to include rejected ORpairs and held nodes in HOR will likely continue to match GSC's performance, and PoD is not much worse. DYN may be desirable for large scale offer networks.

GSC-POD perfroms better than GSC at $p \geqslant 0.7$ without HOR, and worse for the low p. With HOR, GSC-POD performs equally with GSC except that the matched task popularity is an order of magnitude smaller. GSC-POD seems to only succeed in matching hard-to-match tasks much better for low p; in other cases, GSC performs worse than at high p instead of the same. The long term benefits of matching more hard-to-match ORpairs, beside valuing fairness, have not been determined. If the benefits are not significant, GSC-POD is not worth the O(N) step of calculating the PoD for each ORpair.

GSC is simple, fast, and performs well in more conditions than the other matching algorithms. An upgraded DYN may out-do it. GSC-POD's performance suggests there may be beneficial ways of modifying GSC only increasing the linear run-time coefficient [73].

9.2 Feasibility of GSC

GSC is the all-around best performing matching algoirthm, so how does it work as a market mechanism?

Ideally, all ORpairs would be satisfactorily matched with a small wait time. At least, all matcheable ORpairs, as in the graphs tested up to $N_{end} = 15,000$ there were 757 unmatcheable ORpairs[74] This is probably impossible, and even currency fails to satisfy all users in a market. How close can GSC come?

Recall the $N_{initial}$ test with runs of 3000 steps in Section 8.7. Wait time in these runs stabilizes at around 1,100 steps. This is promising, however the total matched also stabilized around 800: Section 8.6 shows the graph size keeps growing over time. That is, if a user's ORpair is matched, it's likely to happen quickly. Otherwise, the wait is long and the number of waiting ORpairs grows. Perhaps this may work with some ORpair expration setting. This, however, with $p = 1$.

73 Moreover, there may be more efficient implementations than used in this thesis

74 Unmatcheable means either the offer or request task had no corresponding ORpairs.

With lower p, theoretically, users will have to be suggested multiple (if not many) matches before getting accepted. Recall that, with HOR, the number is 10 matches prior to acceptance with $p = 0.5$, 3 for $p = 0.7$, 1.5 for $p = 0.9$, and an enourmous 35 rounds for $p = 0.3$. Cursory advice would be not to participate in an offer network if $p < 0.5$, but this seems to be the case for some kidney exchange patient-donor pairs who nonetheless get some help[75].

A promising result for the feasibility of GSC for offer networks is that with HOR and $p = 0.3$, more than 5000 ORpairs are matched in a 5000 step run: whether with $0.3 \leqslant p \leqslant 0.7$ a steady state graph size can be achieved or not needs to be tested. Both wait time and hold wait time are around the stable average value found with $p = 1$. The 2000 held nodes are held for, on average, two rounds[76].

The results indicate that GSC can be feasibly used for offer networks, at mid p ranges matching nearly as many ORpairs as are added with stable, low wait times.

10 FURTHER WORK

The below list covers various additions or fixes to the present implementation or tests to do with it to learn more about dynamic offer networks.

10.1 Upgrade Dynamic Matching

DYN performed poorly as p decreased. This can, with high likelihood, be fixed by adding rejected ORpairs and held ORpairs into the dynamic matching with the newly added ORpairs.

10.2 Cycle Scarcity Analysis

MAX-2 performance hints that 2-cycles are rare. Kidney graphs are dense, but are k-cycles in these scale free graphs rare?

10.3 Fix: Prevent Match Repetition

In the present thesis matching algorithms are permitted to suggest the same match multiple times. With large step-sizes or greedy cycle covers starting at random nodes, this assumption should be ok. However, the good performance of very small step sizes could, in part, be due to repeatedly suggesting

75 Chains and gifts seem to greatly improve the situation for hard-to-match kidneys.

76 This is surprisingly low given the rejection probability of $p = 0.3$. which means that ORpairs are not suggested in matches too many times.

matches until they are eventually accepted. This especially seems a likely explanation of the counterintuitive outcome that more ORpairs are matched with low p than high p; although the results do not seem entirely explained by this. Thickening of the market and increasing dynamism seem to be at play as well.

This fix may require a less concise graph representation such as the one used for MAX-WEIGHT that allow an edge to be removed between one user's `offer` and another user's `request` after one of them rejected the match. The edges could be adde directly between ORpairs.

10.4 ORpair Expiration

Akbarpour [31] found that the greedy strategy was no longer optimal when ORpair expiration time exists and is known to the system. This feature can easily be added by adding an **expire by** property to ORpairs. Weight or greedy search order can be modified to take this property into account.

10.5 Match Suggestion Limits

Especially without HOR, sustained performance at low p relies on suggesting many matches to users prior to acceptance. Human users will likely dislike this. First, the amount of matches suggested each round should be measured, and if it does indeed increase, limits on how many times an ORpair can be matched or with what frequency ORpairs can be matched should be tested.

This could be done by decrementing a counter each time an ORpair is matched. Another method is to remove matched cycles from the graph whether accepted or not, and add them to a queue to be re-added in Q steps.

10.6 Longer Run Experiments

What are the long term implications of an algorithm that matches hard-to-match ORpairs better? Will the number of hard-to-match ORpairs increase with time running another matching algorithm, resulting in a decrease of matches over time?

10.7 Low p Graph Size Experiments

In one 5000 step run more than 5000 ORpairs were matched. Thus graph size over time experiments need to be re-run with this setting: perhaps there is a steady state.

11 ADDITIONAL FEATURES

This section discusses potentially desirable features an an offer network could have that are beyond the scope of this thesis.

11.1 Gift Chains

The impact of a few gifts and their chains will be interesting to investigate. Kidney exchanges increasingly rely on chains [7], and Anderson's theoretical analysis [29] implies the advantage of chains over exchanges only may be comparable to that found by Abbassi for credit mechanisms [32].

11.2 Asynchronous Exchange

Asynchronous exchange as described in the HOR section should be tested.

11.3 Artificial Rejection

Asynchronous exchange proved the offer network with more dynamism. However if mid-p ranges perform better, some suggested cycles could be randomly rejected to provide this benefit if the acceptance probability is too high. Attempting to learn potentials as in Dickerson [7] and exclude these from greedy matching heuristics may also prove fruitful.

11.4 Reputation Biased Matching

As mentioned in the Algorithms section 7, users' reputation can be used to bias matching. This means that users with higher reputation will be more likely to have their ORpairs satisfied.

One way to do this is including reputation in weights. In MAX-WEIGHT, this could be a function of both the offer user's reputation and the request user's reputation.

In GSC, the order of the nodes could be influenced by reputation. Of course shortest weighted paths can be found for GSC, but the function here is limited to $(+)$.

A drawback is that users with lower reputation will have trouble gaining reputaiton as users with higher reputation have more opportunities to further build reputation. A coarse version of this feature would only bias matching against users with no positive feedback yet: this way users are prevented from having to risk agreeing to an exchange with new users.

A modification that counters this drawback would be to bias matching to prefer exchanges with users that have similar reputation: then new users are simply more likely to exchange with each other as they build reputation and test each other.

11.5 Preference Biased Matching

In this thesis' version, acceptance probability is uniform and ORpairs are matched irrespective of the user that uploaded them. However, in practice whether a user accepts a match will depend on the other users' profiles, histories, and reputations. Reputation biased matching is one form of this.

If a user lists knowledgeable categories or skills, then users could upload desirable caterories or skills for users to possess along with an ORpair. These soft contraints[77] could be added in similar to reputation.

Tasks are presently described only by their name. In reality, there may be a hierarchy of types of tasks used for matching. However, if task descriptions are too detailed, then every task may be unique and impossible to match. This means that if there are, for example, 100 offers and requests for a task, some will actually be better matches for each other than others. Hard constraints only define a task insofar as necessary for matching. Additional properties of the specific task desired can be added to the task description to bias the matching, that is, soft constraints.

11.6 Similarity Based Task Grouping

As tasks get more detailed, it may be desirable to match two similar tasks as a user may find this acceptable anyway. This would require some sort of task clustering based on (hierarchical) categories and properties[78]. The desirability may increase with the user's wait time.

Rappaz [33] make headway in this direction with their ordered swap list recommender sytsem.

11.7 ORs and ANDs of offers and requests

11.7.1 *OR*

In Abbassi [23] [32] users just have "wish lists" and "item lists" but they don't specify preferred combinations of items to exchange as ORpairs. This is basically an exclusive OR of offers and requests: (offer: $task_a \vee \cdots \vee$

77 Soft contraints bias matching but do not necessarily need to be satisfied.
78 Viktoras Veitas at the Global Brain Institute deserves credit for this idea.

$task_z$, request: $task_1 \vee \cdots \vee task_{15}$)[79]. A user can specify sets of tasks of subjectively equivalent value. One may want to specify a set of offers one is willing to do any one of in exchange for a request: (offer: $\vee_{t \in Offers} t$, request: $task_a$).

Implementation-wise, one can add an ORpair for each combination: $\{(o, r) : o \in Offers, r \in Requests\}$. However one needs a match to be, in Abbassi's terms [23], *conflict-free*: only one offer and one request in the set is used. This is easy to model in an ILP (integer linear programming) approach (although doing so in an efficient way may be harder). One way in the present graph framework that makes the cycles edge-disjoint is to split the ORpair into two nodes with one edge-between them, the offer node has an edge to each offer and the request node an edge to each request.

11.7.2 *AND*

What if a user needs two separate tasks to be done together (but not only one)? (offer: $task_a$, request $task_b \wedge task_c$). The reason not to make this one task is because different users may do each of the tasks.

In the case of work contracts where one is paid in money, an ORpair with multiple offers that must all be done and one request (money) is very common. Cases in terms of trade of tasks and services alone are harder to think of. This presents a problem as it would be hard to find a cycle with one ORpair requesting two tasks and none offering two, unless there are task that can satisfy multiple users.

At present, the author is not sure how to implement ORpairs with ANDs in the graph framework.

11.8 Conditional Offers and Requests

One theme discussed a lot with the Global Brain Institute is whether an offer network can be used to organize work on open source projects. Thus one may want to offer to work on a project if and only if 4 other people offer to work on the project.

The distinguishing feature is that the 'request' in a conditional offer does not need to be part of a cycle or an exchange. However, the conditional event (task being done) could be part of a cycle involving the conditional offer being done!

At present, the author is not sure how to implement conditional ORpairs.

11.8.1 *Decentralized Offer Network*

In practice an offer network could easily become big enough that decentralization is necessary. The results in this thesis, Anderson [29], and Jia [22]

79 $\vee$:- OR and $\wedge$:- AND

imply that some variant of greedy cycle covering will suffice. Distributed cycle detection has been studied significantly for deadlock detection. Thus decentralized offer network design seems promising.

12 CONCLUSION

There are many domain-specific barter exchange sites, and a few general ones. The experiments in this thesis suggest that a fast heuristic matching algorithm (GSC with HOR) can overcome the difficulty in bartering: arranging k-user exchanges ($k > 2$), even when the probability users accept matches is not high. The wait time calculated in the thesis depends on how many users ask for exchanges, so a real life offer network that does not reach critical mass will be unbearably slow (however, this is a problem for any social network start-up). Unlike algorithms that find maximal solutions on the whole network, a GSC variant can easily be extended to a decentralized offer network, which is also promising.

Currency allows anything to be exchange for anything, provided there is price parity. This parity, price, is typically understood to be set globally in the marketplace by negotiating to buy for the lowest price possible or sell for the highest price possible, and these prices are influenced by the supply and demand in the market. In the offer network's money-less paradigm, there must be direct evidence for the parity of tasks in an exchange, regardless of the dynamics of the rest of the offer network. This feature is important. First, because the exchange value of a task/good is more concrete: the set of tasks in cycles with $\mathtt{task}_a$ are, ostensibly, of equal value. Second, users may have (subjective) mutually exclusive classes of tasks they are willing to exchange. For example, an author may exchange a book review for another favor but not for money (however will work for money in other ways). Another example is, of course, kidney exchange: they are not allowed to be exchanged for money, or nearly anything other than kidneys (or other organs). Thus the direct evidence of task parity in offer networks allows different classes of tasks to be exchanged in one market.

The counter-intuitive performance gains with moderate acceptance probability point toward directions for improvement, all of which increase the ways in which ORpairs can be matched. Increasing the ways in which ORpairs can be matched makes an offer network's performance more like that of a currency-based market. Abbassi [32] try to do this by modeling asynchronous exchange with a credit system; however this becomes another form of money and loses the above feature. When HOR is used, an ORpair can facilitate one match and then become part of a held ORpair to facilitate another match. This allows more flexibility in exchanges while maintaining evidence of parity. The method of using waiting priority queues for pure **requests** and gift chains to model asynchronous exchange decribed in Section 7.5.3, a generalization of HOR, maintains parity and allows more types of exchanges as gifts are, in essence, treated like one unit of task-typed currency. Similarity based task grouping and ORpairs with ORs of offers and

requests also make more matches possible. The potentials method of Dickerson [7] tries to predict which parts of the offer network will be more useful later and save them: while greedy matching is good, matching everything possible greedily produces poorer results. Even doing this randomly, via low acceptance probability, seems according to this thesis' experiments benefitial.

The experiments in this thesis indicate that GSC with HOR makes offer networks feasible, with respect to total matched ORpairs and the average wait time. Further experimentation is advised. Moreover, there are indications significant improvements can be made by improving the model used for matching in the offer network, much as Dickerson [7] [26] and Jia [22] found that approximation algorithms that take acceptance probability into account can out-perform optimal algorithms that don't, and as MAX-WEIGHT performs poorly without HOR.

---

13 REFERENCES

REFERENCES

[1] D. Graeber. *Debt: The First 5000 Years*. Penguin Books Limited, 2012.

[2] David J. Abraham, Avrim Blum, and Tuomas Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, EC '07, pages 295–304, New York, NY, USA, 2007. ACM.

[3] Valentin Katasonov. Moneyless business: Barter transactions increase worldwide. https://www.strategic-culture.org/news/2016/02/21/moneyless-business-barter-transactions-increase-worldwide.html, 2016. Accessed: 2017-06-26.

[4] Mark Milian. Data bartering is everywhere. https://www.bloomberg.com/news/articles/2012-11-15/data-bartering-is-everywhere, 2012. Accessed: 2017-06-26.

[5] Wenyi Fang, Pingzhong Tang, and Song Zuo. Digital good exchange. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 264–270. AAAI Press, 2016.

[6] Alvin E. Roth. Repugnance as a constraint on markets. Working Paper 12702, National Bureau of Economic Research, November 2006.

[7] John P. Dickerson. *A Unified Approach to Dynamic Matching and Barter Exchange*. PhD thesis, Carnegie Mellon University, Sep 2016.

[8] Y. Beyene, M. Faloutsos, D. H. Chau, and C. Faloutsos. The ebay graph: How do online auction users interact? In *IEEE INFOCOM Workshops 2008*, pages 1–6, April 2008.

[9] Ben Goertzel, Ted Goertzel, and Zarathustra Goertzel. The global brain and the emerging economy of abundance: Mutualism, open collaboration, exchange networks and the automated commons. *Technological Forecasting and Social Change*, 114:65 – 73, 2017.

[10] Ben Goertzel. Beyond money: Offer networks, a potential infrastructure for a post-money economy. In Ben Geortzel and Ted Goertzel, editors, *The End of the Beginning: Life, Society and Economy on the Brink of the Singularity*, page 522–549. Humanity+ Press, 2015.

[11] Francis Heylighen. The offer network protocol: Mathematical foundations and a roadmap for the development of a global brain. *The European Physical Journal Special Topics*, 226(2):283–312, 2017.

[12] Francis Heylighen. Towards an intelligent network for matching offer and demand: From the sharing economy to the global brain. *Technological Forecasting and Social Change*, 114:74 – 85, 2017.

[13] Florian Kleedorfer, Christina Maria Busch, Christian Pichler, and Christian Huemer. The case for the web of needs. In *Proceedings of the 2014 IEEE 16th Conference on Business Informatics - Volume 01*, CBI '14, pages 94–101, Washington, DC, USA, 2014. IEEE Computer Society.

[14] Rizzi Biro, Manlove. Maximum weight cycle packing in directed graphs, with application to kidney exchange programs. *Discrete Mathematics, Algorithms and Applications, volume 1, number 4*, pages 499–517, 2009.

[15] Alvin Roth, M. Utku Ãœnver, and Tayfun SÃ¶nmez. Kidney Exchange. Scholarly Articles 2580565, Harvard University Department of Economics, 2004.

[16] Alvin E. Roth, Tayfun Sönmez, and M. Utku Ünver. Efficient kidney exchange: Coincidence of wants in markets with compatibility-based preferences. *American Economic Review*, 97(3):828–851, June 2007.

[17] Segev DL, Gentry SE, Warren DS, Reeb B, and Montgomery RA. Kidney paired donation and optimizing the use of live donor organs. *JAMA*, 293(15):1883–1890, 2005.

[18] Tyler Greer. Donors help extend uab kidney chain to record 51 transplants. https://www.uabmedicine.org/-/donors-help-extend-uab-kidney-chain-to-record-51-transplants, 2013. Accessed: 2017-06-22.

[19] Ross Anderson, Itai Ashlagi, David Gamarnik, and Alvin E. Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112(3):663–668, 2015.

[20] Kristiaan M. Glorie, J. Joris van de Klundert, and Albert P. M. Wagelmans. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management*, 16(4):498–512, 2014.

[21] Benjamin Plaut, John P. Dickerson, and Tuomas Sandholm. Fast optimal clearing of capped-chain barter exchanges. In *Proceedings of the Thirtieth*

AAAI Conference on Artificial Intelligence, AAAI'16, pages 601–607. AAAI Press, 2016.

[22] Zhipeng Jia, Pingzhong Tang, Ruosong Wang, and Hanrui Zhang. Efficient near-optimal algorithms for barter exchange. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 362–370, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[23] Laks V. S. Lakshmanan and Zeinab Abbassi. On efficient recommendations for online exchange markets. *2009 IEEE 25th International Conference on Data Engineering. ICDE 2009*, 00:712–723, 2009.

[24] John P. Dickerson, David F. Manlove, Benjamin Plaut, Tuomas Sandholm, and James Trimble. Position-indexed formulations for kidney exchange. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, pages 25–42, New York, NY, USA, 2016. ACM.

[25] John P. Dickerson, Aleksandr M. Kazachkov, Ariel D. Procaccia, and Tuomas Sandholm. Small representations of big kidney exchange graphs. *CoRR*, abs/1605.07728, 2016.

[26] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. In *Proceedings of the Fourteenth ACM Conference on Electronic Commerce*, EC '13, pages 323–340, New York, NY, USA, 2013. ACM.

[27] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, pages 507–523, Berlin, Heidelberg, 2011. Springer-Verlag.

[28] John P. Dickerson and Tuomas Sandholm. Multi-organ exchange: The whole is greater than the sum of its parts. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 1412–1418. AAAI Press, 2014.

[29] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. A dynamic model of barter exchange. In *Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1925–1933, Philadelphia, PA, USA, 2015. Society for Industrial and Applied Mathematics.

[30] P. Erdős and A Rényi. On the evolution of random graphs. In *PUBLICATION OF THE MATHEMATICAL INSTITUTE OF THE HUNGARIAN ACADEMY OF SCIENCES*, pages 17–61, 1960.

[31] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. *CoRR*, abs/1402.3643, 2014.

[32] Zeinab Abbassi, Laks V. S. Lakshmanan, and Min Xie. Fair recommendations for online barter exchange networks. In Angela Bonifati and

Cong Yu, editors, *Proceedings of the 16th International Workshop on the Web and Databases 2013, WebDB 2013, New York, NY, USA, June 23, 2013.*, pages 43–48, 2013.

[33] Jérémie Rappaz, Maria-Luiza Vladarean, Julian McAuley, and Michele Catasta. Bartering books to beers: A recommender system for exchange platforms. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, pages 505–514, New York, NY, USA, 2017. ACM.

[34] Brian F. Cooper and Hector Garcia-Molina. Peer-to-peer data trading to preserve information. *ACM Trans. Inf. Syst.*, 20(2):133–170, April 2002.

[35] Kostas G. Anagnostakis and Michael Greenwald. Exchange-based incentive mechanisms for peer-to-peer file sharing. In *24th International Conference on Distributed Computing Systems (ICDCS 2004), 24-26 March 2004, Hachioji, Tokyo, Japan*, pages 524–533. IEEE Computer Society, 2004.

[36] D. Cabanillas. *Peer-to-Peer Bartering: Swapping Amongst Self-interested Agents*. PhD thesis, Universitat Politècnica de Catalunya, Apr 2009.

[37] David Cabanillas and Steven Willmott. Self-organization amongst non-altruistic agents for distribution of goods: Comparing bartering and currency based exchange. In *EUMAS*, 2006.

[38] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[39] Béla Bollobás, Christian Borgs, Jennifer T. Chayes, and Oliver Riordan. Directed scale-free graphs. In *SODA*, pages 132–139. ACM/SIAM, 2003.

[40] Bodo Manthey. Minimum-weight cycle covers and their approximability. *CoRR*, abs/cs/0609103, 2006.

[41] Qiang Ye, Min Xu, Melody Kiang, Weifang Wu, and Fangfang Sun. In-depth analysis of the seller reputation and price premium relationship: A comparison between ebay us and taobao china. *Journal of Electronic Commerce Research*, 14(1):1–10, 2013.

[42] Atila Abdulkadiroğlu and Tayfun Sönmez. House allocation with existing tenants. *Journal of Economic Theory*, 88(2):233–260, 1999.

14 APPENDIX: SUPPLEMENTAL MATERIALS

14.1 Run Time Plots

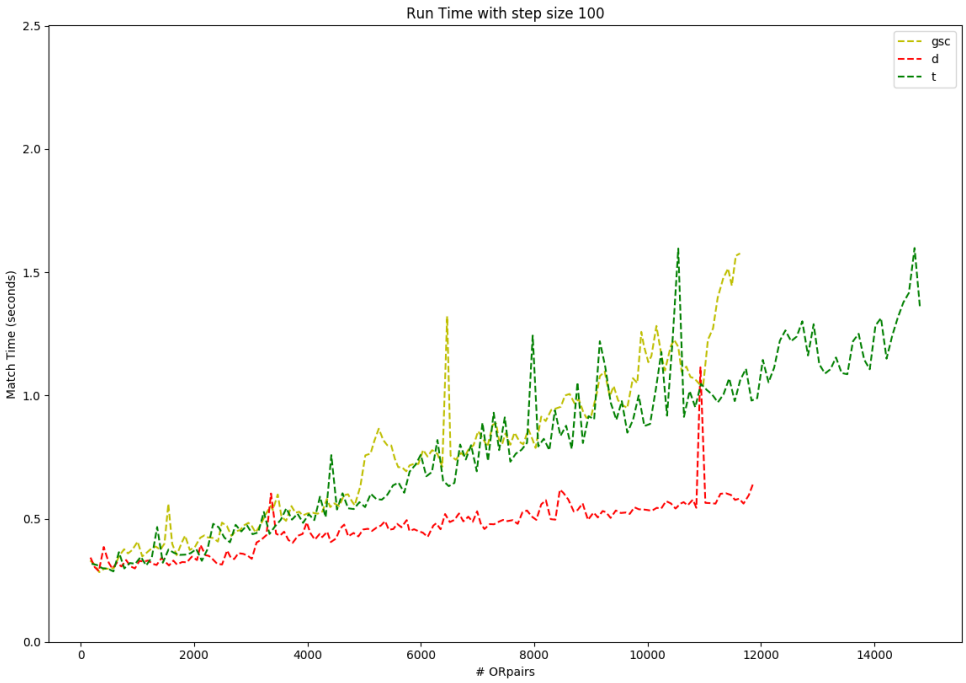


Figure 47: BM1 - Run Times - Step Size 100

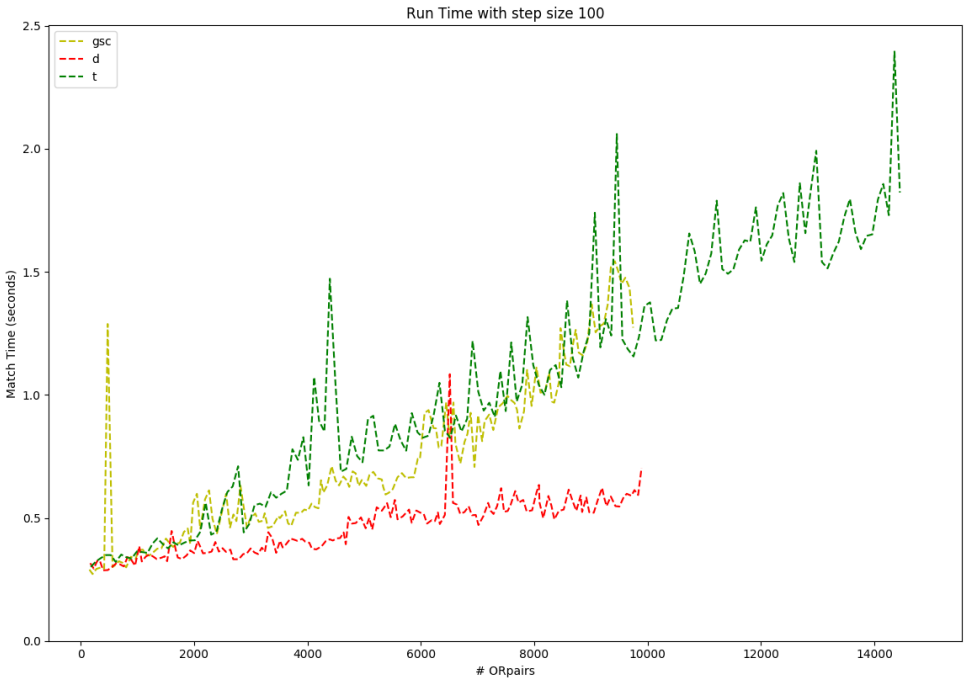


Figure 48: RB1 Run Times - Step Size 100

14.2 Step Size Test Plots

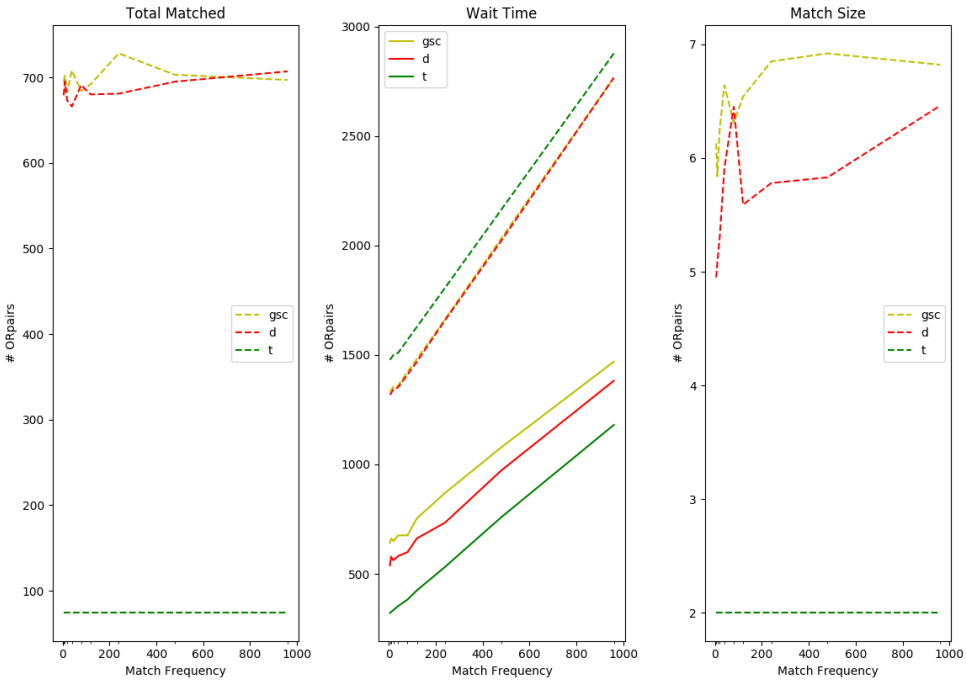


Figure 49: EN - Step Size Test 1

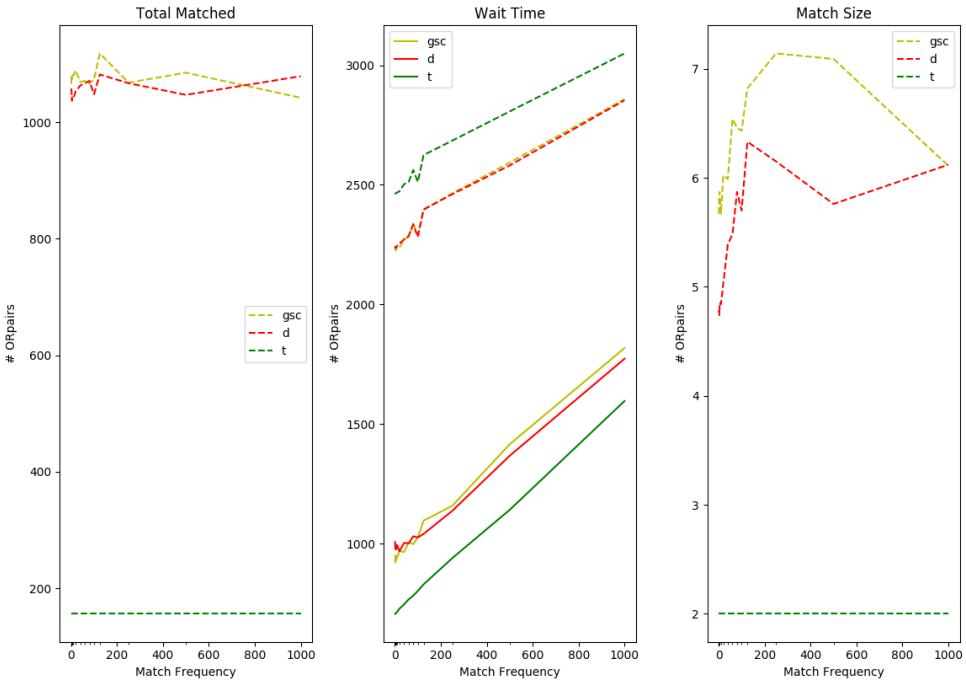


Figure 50: BM1 - Step Size Test 2

14.3 Ninitial Test Plots

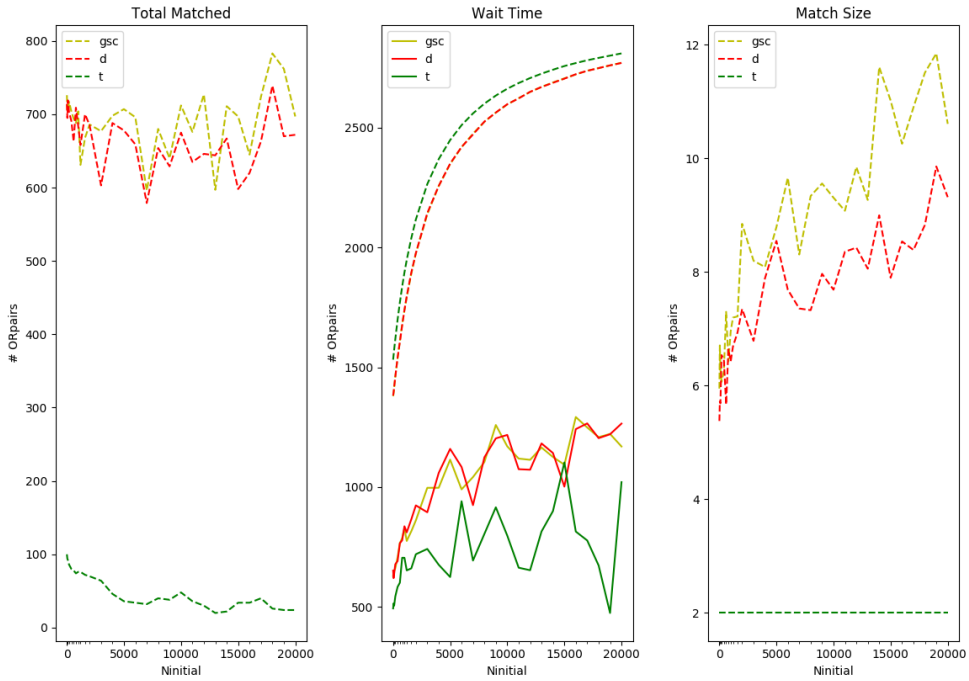


Figure 51: Ninitial Test - BM1

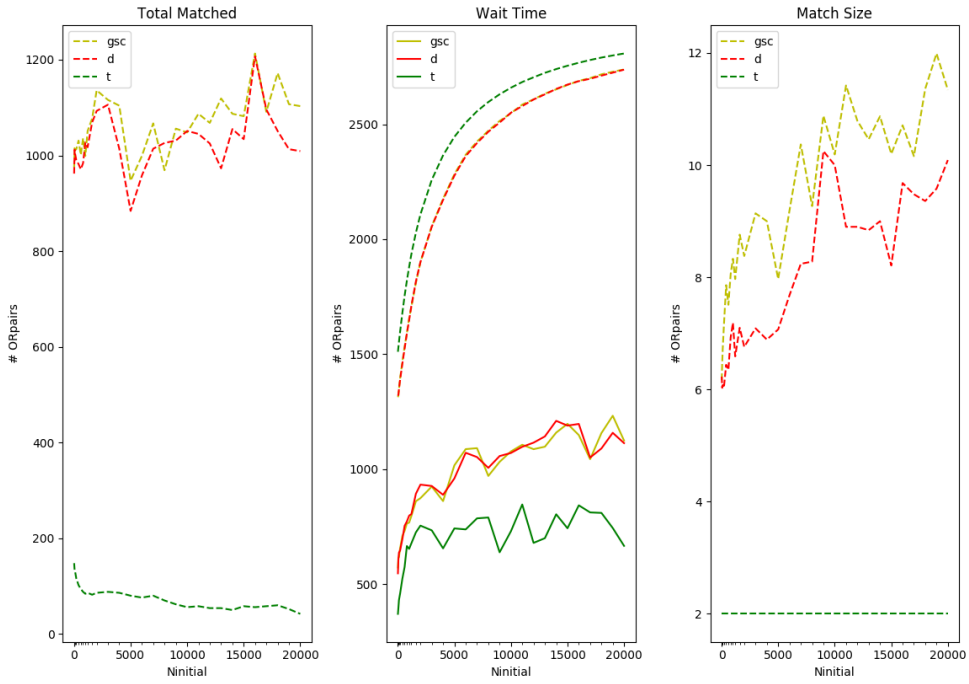


Figure 52: Ninitial Test - RB1

14.4 Performance Consistency Test

Below are four experiment batches from Section 8.9 to check that the results do not depend on particulars of RB1:

1)

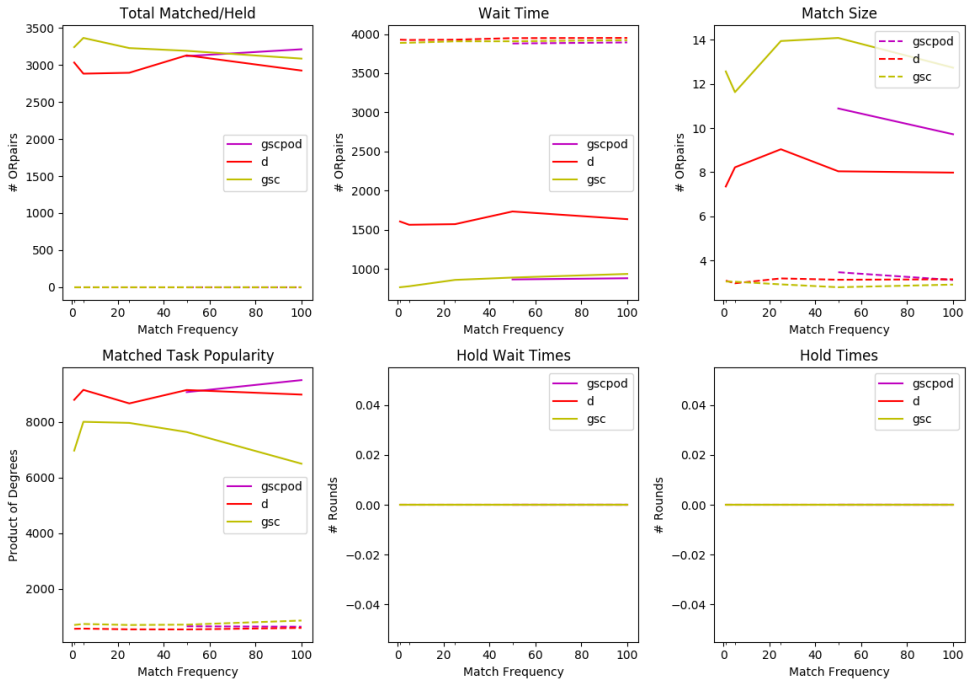


Figure 53: Performance Test - RB1 - p = 0.9 without HOR

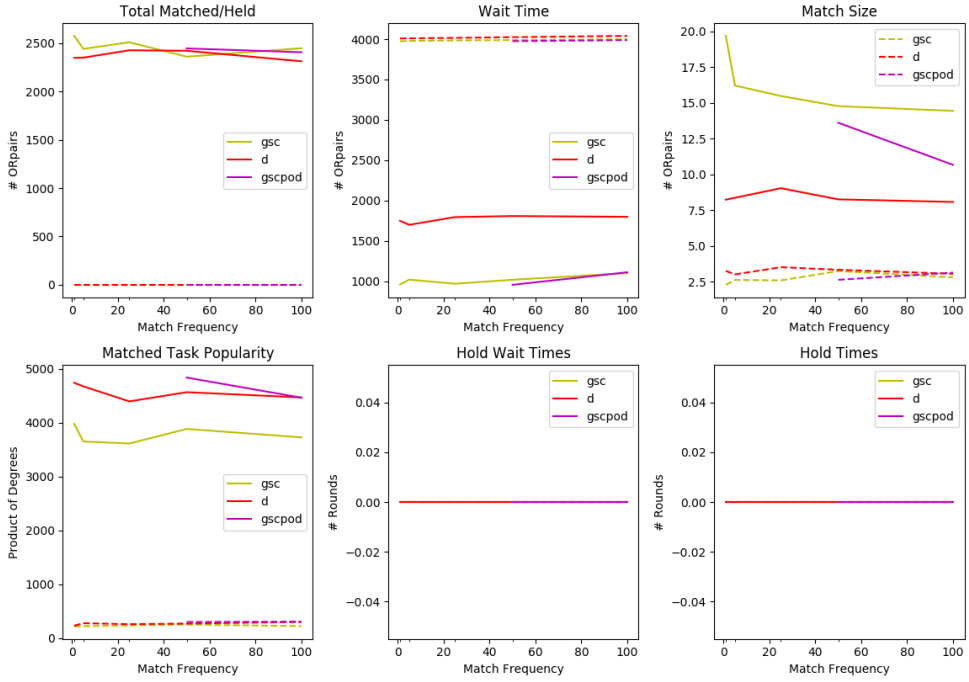


Figure 54: Performance Test - EN - p = 0.9 without HOR

2)

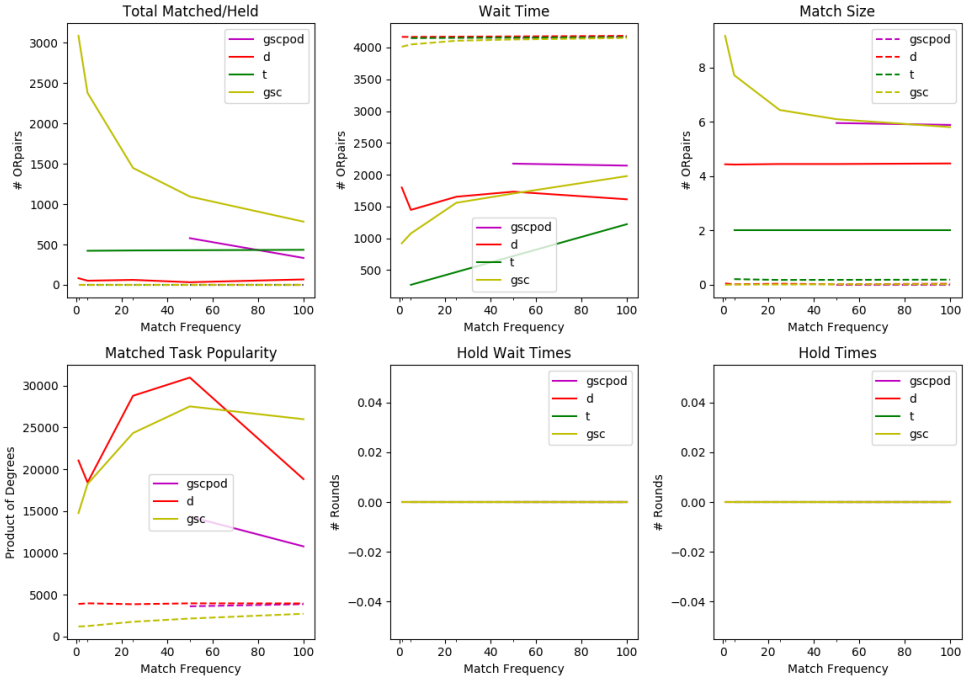


Figure 55: Performance Test - RB1 - p = 0.3 without HOR

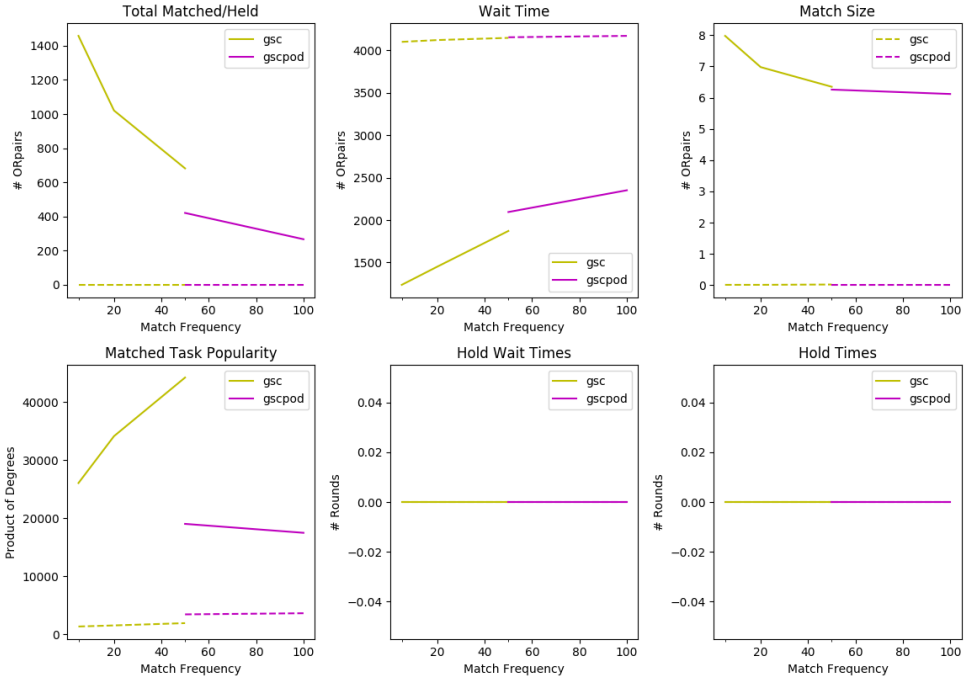


Figure 56: Performance Test - BM1 - p = 0.3 without HOR

3)

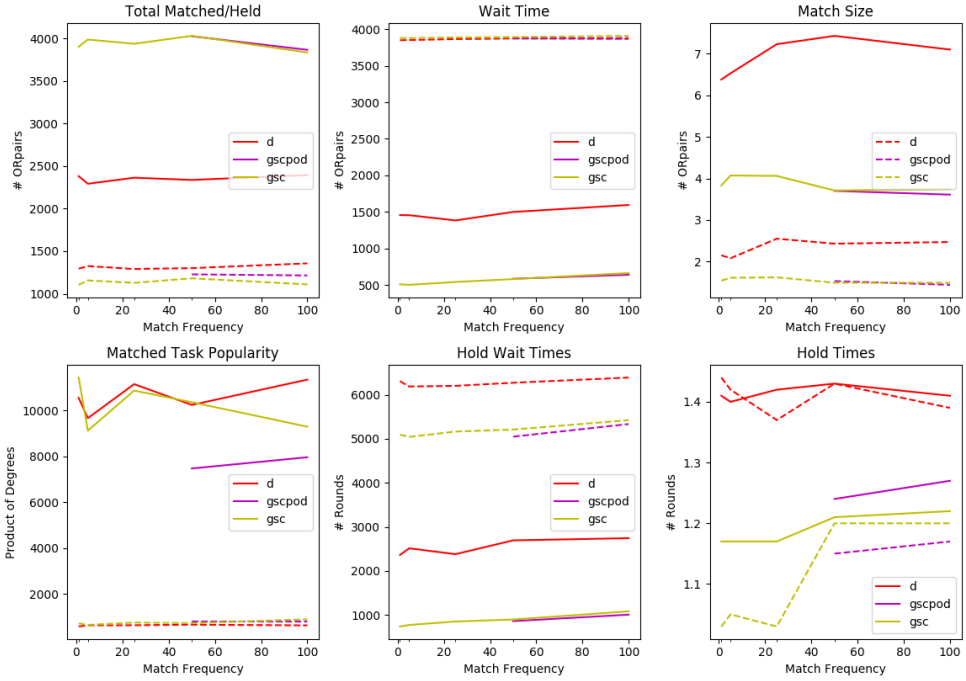


Figure 57: Performance Test - RB1 - p = 0.7 with HOR

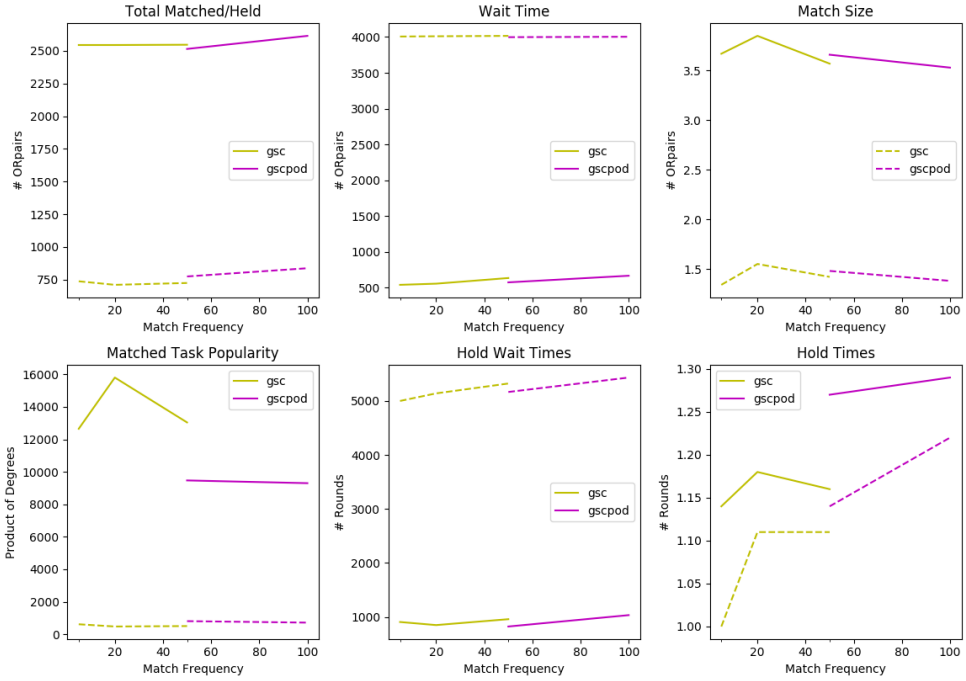


Figure 58: Performance Test - BM1 - p = 0.7 with HOR

4)

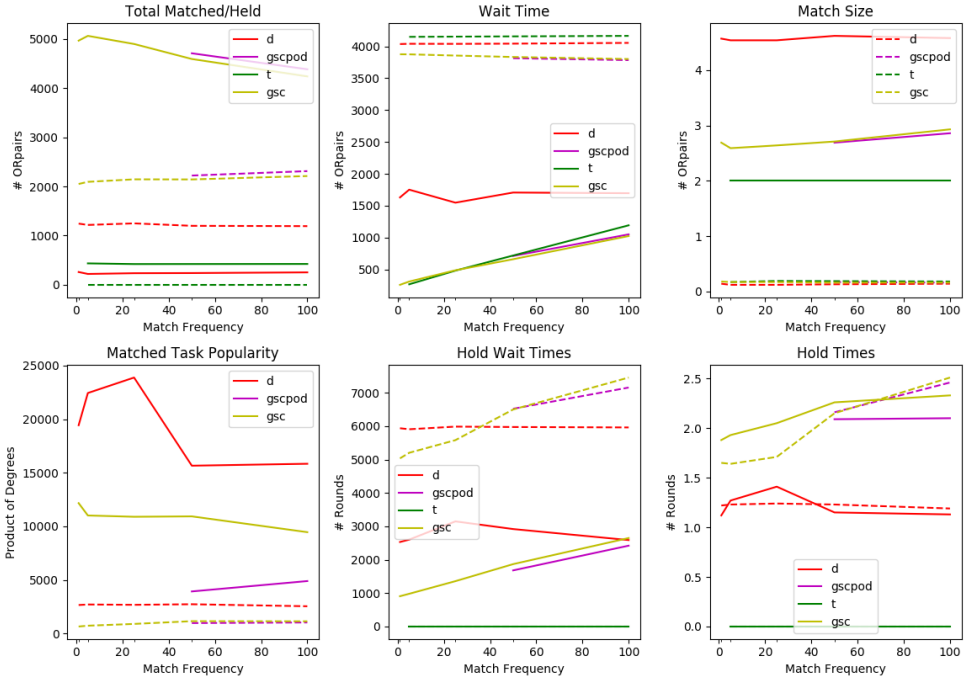


Figure 59: Performance Test - RB1 - p = 0.3 with HOR

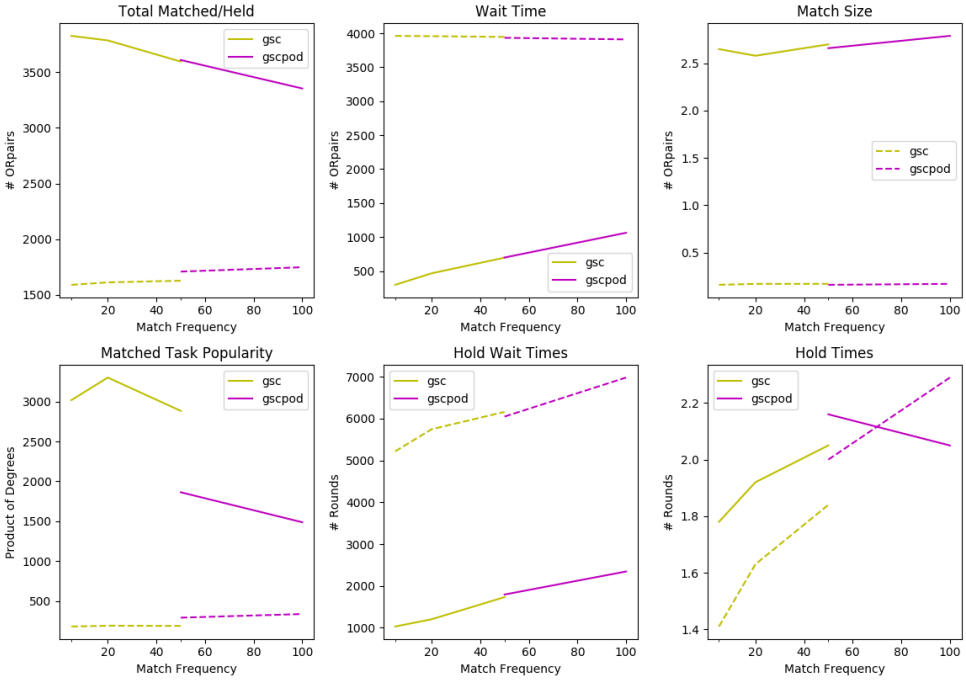


Figure 60: Performance Test - EN - p = 0.3 with HOR