Extending multi-agent pathfinding towards item transportation, with comparisons between heuristics in different settings

Swedish title: Vägplanering för agenter med utvidgning mot objekttransport, med jämförelser mellan heuristiker i olika omgivningar

ANTON HAMPUS

DENNY LEKIC
Extending multi-agent pathfinding towards item transportation, with comparisons between heuristics in different settings

ANTON HAMPUS, DENNY LEKIC
Abstract

In an increasingly connected world problems of planning transport of different kinds is becoming increasingly important. This study has proposed an extension of the multi-agent pathfinding problem to include item transportation, as well as providing possible cost-based heuristics with accompanying comparisons. The aim was to explore the usefulness of the heuristics together with how the quality of their solutions depend on differences in the setting. The results are promising for two cost metrics explored that have shown benefits when adding them to a simple base heuristic. The study also indicates patterns in which the makespan of the solutions depend on the problem instance characteristics.
Sammanfattning

# Contents

1 Introduction .......................... 1
   1.1 Purpose and goal ...................... 1
   1.2 Outline .................................. 1
   1.3 Predecessors of MATIP ................. 2
      1.3.1 Non-anonymous multi-agent pathfinding ....... 2
      1.3.2 Anonymous multi-agent pathfinding .......... 2
      1.3.3 Target assignment and path finding (TAPF) .... 3
   1.4 Multi-agent item transportation problem (MATIP) .... 3
      1.4.1 Input ........................................ 3
      1.4.2 Objective ..................................... 3
   1.5 Research question ..................... 5
   1.6 Scope ....................................... 5

2 Conceptual foundation ................. 6
   2.1 Simplifying the real world ............ 6
   2.2 Centralized planning ................... 6
   2.3 Decoupled planning ..................... 7
   2.4 A* search algorithm .................... 7
   2.5 Conflict-based search .................. 7
   2.6 Batch processing ....................... 7
   2.7 Real-time processing ................... 8

3 Methods .................................. 9
   3.1 Terminology ............................. 9
      3.1.1 Solutions and optimal solutions ............. 9
      3.1.2 Makespan .................................... 9
      3.1.3 Checkpoints ................................ 9
      3.1.4 Individual predicted makespan ............... 9
      3.1.5 Relative predicted makespan .................. 10
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.6 Global predicted makespan</td>
<td>10</td>
</tr>
<tr>
<td>3.1.7 Makespan prediction error</td>
<td>10</td>
</tr>
<tr>
<td>3.1.8 Relative makespan prediction error</td>
<td>10</td>
</tr>
<tr>
<td>3.2 Solving MAITP</td>
<td>10</td>
</tr>
<tr>
<td>3.3 Heuristics used</td>
<td>11</td>
</tr>
<tr>
<td>3.4 Experiments</td>
<td>13</td>
</tr>
<tr>
<td>4 Results</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Cost metrics</td>
<td>15</td>
</tr>
<tr>
<td>4.2 Number of agents</td>
<td>17</td>
</tr>
<tr>
<td>4.3 Agent carrying capacity</td>
<td>19</td>
</tr>
<tr>
<td>4.4 Drop-offs</td>
<td>21</td>
</tr>
<tr>
<td>4.5 Edges</td>
<td>23</td>
</tr>
<tr>
<td>4.6 Items</td>
<td>25</td>
</tr>
<tr>
<td>5 Discussion</td>
<td>28</td>
</tr>
<tr>
<td>6 Conclusion</td>
<td>30</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The rapid development of robot technology has made a plethora of things possible; everything ranging from military swarm-drones to warehouse automation. Questions such as how to distribute workload efficiently and what factors increase efficiency are of growing importance as industries such as e-commerce grow. This study will focus on planning efficient paths for agents in graphs where agents can pick up and carry multiple items, as well as drop them off. A practical implementation would be a warehouse-like setting and an interesting aspect is finding an optimal solution. However, often optimal solutions take too long and so this study explores heuristics as a complementary solution.

1.1 Purpose and goal

The purpose of this study is to propose a new optimization problem with more explicit compliance with areas such as robotic warehouses, autonomous taxi services and others of similar nature. The study also intends to examine what heuristics may be useful in solving the problem and how the quality of their solutions depend on differences in the setting.

1.2 Outline

First similar predecessors of the proposed problem are summarized, followed by a formal definition of the new problem itself (referred to as the multi-agent item transportation problem, or MAITP for short) together with the research question and the scope of the study. Thereafter follows a description of the examined heuristics and the experiments conducted with them. The results from
these experiments are then summarized and analyzed, whereupon conclusions are drawn and future research is suggested.

1.3 Predecessors of MAITP

Previous research has been conducted on variations of the so-called multi-agent pathfinding problem (MAPF problem) and we propose an extension to this problem. The following research will help the reader understand the natural progression of the problem and why our extension is a possible next step.

1.3.1 Non-anonymous multi-agent pathfinding

One variant of the multi-agent pathfinding (MAPF) problem is defined in the following way, consistent with [2, 7]:

Given a graph \( G = (V, E) \) and \( k \) agents \( a_1, a_2, ..., a_k \), where each agent \( a_i \) has associated start and goal vertices \( s_i \) and \( g_i \). At time \( t = 0 \), each agent \( a_i \) is located on vertex \( s_i \). Before every positive time value, each agent is allowed to traverse at most one edge (to a neighboring vertex). Every vertex can contain at most one agent at a time, and every edge can be traversed by at most one agent at a time (meaning no collisions are allowed to happen). The problem is to find a sequence of legal moves for each agent that together minimize a given objective function, and such that there exists a time where every agent \( a_i \) is located on vertex \( g_i \) simultaneously.

We will refer to this version as the non-anonymous multi-agent pathfinding problem. By reducing the sliding tile puzzle problem for example [2], it can be proven that non-anonymous MAPF is NP-complete, given that the objective function is the minimum time it takes to get all agents to their respective goal vertices (the makespan) [4]. Approaches used to deal with the problem include for example conflict-based search or reductions to satisfiability problems, integer linear programming or answer set programming [4].

1.3.2 Anonymous multi-agent pathfinding

Another variant of the multi-agent pathfinding problem involves having \( k \) agents, each with an associated starting vertex, but with the modification that any agent can choose any goal vertex, with the constraint that no two agents pick the same one. Unlike the non-anonymous version, the anonymous MAPF problem is solvable in polynomial time, for example by using a maximum flow algorithm [4].
1.3.3 Target assignment and path finding (TAPF)

In keeping with the terminology and notation of [4], the (general) combined target assignment and pathfinding (TAPF) problem will refer to a mixture of the non-anonymous and anonymous MAPF problems stated above. More specifically, inputs include a graph $G = (V, E)$, $K$ teams $t_1, t_2, ..., t_K$ where each team $t_i$ has $K_i$ agents $a_{i1}, a_{i2}, ..., a_{iK_i}$. Also associated with each team $t_i$ is a set of $K_i$ goal vertices $g_{i1}, g_{i2}, ..., g_{iK_i}$, where any agent within each team can choose any of the goal vertices associated with that team (still with the constraint that no two agents can pick the same goal). Every agent $a_{ij}$ has a specified start position $s_{ij}$ as before. In comparison with the previous two MAPF variants, the problem resembles the non-anonymous problem on a global level (each team has a unique set of goal vertices) and the anonymous problem on a team level (each agent within a given team can be assigned to any of the team’s goal vertices). The rest of the problem definition (the movement and time constraints as well as the objective) are the same for each of the three versions of the problem. The non-anonymous problem is thus a special case of the general TAPF problem in which each team has size 1, while the anonymous variant is a spacial case where there is only one team. TAPF can be solved optimally, for example by using the Conflict-Based Min-Cost-Flow algorithm presented by [4].

1.4 Multi-agent item transportation problem (MAITP)

1.4.1 Input

Given is an undirected graph $G = (V, E)$, a set $D \subseteq V$ of drop-off points and a set $O$ of $l$ items \( \{o_j\}_{j=1}^l \), where each item $o_j$ has an associated weight $w_j \in \mathbb{R}_{>0}$ and initial position $p_j \in V$. Also given is a set of $k$ agents $\{a_i\}_{i=1}^k$, each $a_i$ having an associated start vertex $s_i \in V$ and carrying capacity $c_i \in \mathbb{R}_{>0}$.

1.4.2 Objective

With input as stated above, the problem is to find a sequence of vertices (a path) and a sequence of sets of items (a time-extended inventory) for each agent $a_i$, i.e. to find sequences
$\Gamma_i = \left( \gamma_i^{(t)} \right)_t^{T_{i=0}}$ and $\Xi_i = \left( \xi_i^{(t)} \right)_t^{T_{i=0}}, i = 1, 2, \ldots, k,$

that minimize $T > 0$ and where $\forall i \forall t \left( \gamma_i^{(t)} \in V \land \xi_i^{(t)} \subseteq O \right)$, subject to the following constraints (where $\forall t < T$ and $\exists t < T$ mean the same as the typical notations $\forall t < T$ and $\exists t < T$, respectively):

1. Every agent starts at its specified starting position, i.e.
   $$\forall i \gamma_i^{(0)} = s_i$$

2. All inventories are initially empty, i.e.
   $$\forall i \Xi_i^{(0)} = \emptyset$$

3. At each time unit, an agent can either stay idle or move to a neighboring vertex, i.e.
   $$\forall i \forall t < T \left( \gamma_i^{(t)} = \gamma_i^{(t+1)} \lor \left( \gamma_i^{(t)}, \gamma_i^{(t+1)} \right) \in E \right)$$

4. No two agents can occupy the same position at the same time, i.e.
   $$\forall i \forall i' \forall t \left( \gamma_i^{(t)} = \gamma_i'^{(t)} \rightarrow i = i' \right)$$

5. No two agents can cross the same edge at the same time (they can not meet in the middle), i.e.
   $$\forall i \forall i' \forall t < T \left( \left( \gamma_i^{(t)} = \gamma_i'^{(t+1)} \land \gamma_i^{(t)} = \gamma_i'^{(t+1)} \right) \rightarrow i = i' \right)$$

6. An agent can pick up an item only if they share the same position, i.e.
   $$\forall i \forall j \forall t < T \left( o_j \in \xi_i^{(t+1)} \setminus \xi_i^{(t)} \rightarrow p_j = \gamma_i^{(t+1)} \right)$$

7. No agent can carry more than its carrying capacity, i.e.
   $$\forall i \forall t \left( \sum_{j: o_j \in \xi_i^{(t)}} w_j \leq c_i \right)$$
(8) Every item is picked up exactly once, i.e.
\[ \forall j \exists i \exists t \left( o_j \in \xi_i^{(t+1)} \setminus \xi_i^{(t)} \land \forall i' \forall t' < T \left( i' \neq i \lor t' \neq t \rightarrow o_j \notin \xi_i^{(t' + 1)} \setminus \xi_{i'}^{(t')} \right) \right) \]

(9) Agents can only drop off items on drop-off points, i.e.
\[ \forall i \forall t < T \left( \xi_i^{(t)} \setminus \xi_i^{(t + 1)} \neq \emptyset \rightarrow \gamma_i^{(t + 1)} \in D \right) \]

(10) Every item is dropped off at a time less than or equal to \( T \), i.e.
\[ \forall j \exists i \exists t < T \left( o_j \in \xi_i^{(t)} \setminus \xi_i^{(t + 1)} \right) \]

### 1.5 Research question

How can heuristics be used to solve the multi-agent item transportation problem and how do the quality of their solutions depend on characteristics of the problem instances?

### 1.6 Scope

This study a simplified model of the real world and also has certain restrictions. The only heuristics investigated are cost-based heuristics and the problem includes a variety of characteristics that vary. The varying characteristics of the problem instances are limited to: number of edges, number of drop-offs, number of items, number of agents, and agent capacity. Moreover, in all test instances, the agent capacities are homogeneous and item weights are all equal.

This above mentioned factors impact the generalizability of the study but the study still has a whole variety of planning applications where a homogeneous environment exists. For instance, in areas where weights are less of a problem, and all robots have the same specifications, the study is highly relevant.
Chapter 2

Conceptual foundation

2.1 Simplifying the real world

In order to make the process of planning more efficient, a simplified environment can be assumed instead of modelling the real world as closely as possible. For instance, the warehouse might be represented using a general planar graph or a grid-based planar graph, time might be discretized and the robots might be simplified into points that traverse the graph one edge at a time. Throughout this report, these simplified models of the robots will be referred to as agents. Because these assumptions may be unrealistic, the plans derived using them might not be directly applicable in practice. Instead, the primitive plans are often post processed to construct simple temporal networks that take into account kinematic constraints and distance. These temporal networks guarantee a safety distance and allow for better execution in the real environment [5].

2.2 Centralized planning

A centralized approach to multi agent systems entails one source of control that plans what each individual agent will do. In theory, this seems like an ideal approach to some environments where there is little to no disturbances like exogenous events and where most information seems to be available up-front [3]. In practice, however, robots may have to interact with exogenous events that very much are unpredictable, examples of these include: people, wind, water spills, breakdowns & machine failure etc. The centralized approach is thus something desirable, but in many cases, hard to achieve. The problems are thus practical and it lies in the problem that many things are too complex (i.e have too many independent variables to take into account) and are thus
hard to model accurately.

2.3 Decoupled planning

As opposed to coupled planning, this case is implemented by planning routes for each robot individually. That is, instead of planning a step-by-step solution for all robots simultaneously, the solutions is to plan for each robot in succession and accordingly avoid conflicts [6].

In general decoupled approaches find a path for agents $1...n$ in order and adjust each successive agent based on a global reservation table that is constantly updated. Thus, the successive agents must avoid collisions, wait at certain spots, and so on if need be. Other approaches include establishing traffic laws, segmenting areas etc [6].

2.4 A* search algorithm

The A* algorithm is commonly used pathfinding and traversal algorithm. It is a greedy algorithm that allows revisiting nodes (i.e best-first) and chooses the path that appears to lead to the best solution. The algorithm is based on Dijkstra’s algorithm and uses a heuristic that estimates a cost for each path (the path that appears to be the best), which is tries to minimize [1].

2.5 Conflict-based search

Conflict-based search (CBS) is a two-level algorithm for the MAPF problem that guarantees optimal solutions. At the low level it plans optimal paths for the agents independently given time and position constraints, and at the high level it searches a binary search tree where each node consists of solutions for the agents and constraints are added to sub nodes if the solutions conflict [6].

2.6 Batch processing

Batch processing is a method of processing planning of robot tasks among other things and is as the name entails done in batches. Imagine the warehouse of a retailer who obtains orders throughout the day and perhaps the orders slow down significantly over night. Instead of each robot picking up the orders as they come, and thus doing it quite inefficiently, the retailer waits for a little
while and tries to optimize a route for the robots overnight where they can pick up multiple packages along the way.  

2.7 Real-time processing

The idea of batch processing is to be efficient but has recently faced some critique as the belief is to stay as lean as possible (in terms of inventory). Kiva states that in dynamic environments, and with modern day tools, real-time processing is favourable. However, there are problems with real-time processing as well. Examples include: shelf space, allocation, physical space, and robot allocation.
Chapter 3

Methods

3.1 Terminology

3.1.1 Solutions and optimal solutions
In the following chapters, any set of paths for a MAITP instance that follows constraints 1 – 10 in the problem definition is regarded as a solution to the problem, even if it is not the shortest solution possible. Any such solution that is also as short as the shortest possible solution is called an optimal solution.

3.1.2 Makespan
Given a set of paths, i.e. sequences of vertices, we define the makespan to be the number of steps taken in the longest such path. For example, in any solution to the MAITP, all paths are equally long and thus the makespan is equal to the length of any of the agents’ paths, minus one. If the solution is optimal, the makespan is equal to $T$ from the problem definition of MAITP.

3.1.3 Checkpoints
Given a graph and an agent, we define checkpoints to be ordered vertices that the agent must visit in succession.

3.1.4 Individual predicted makespan
Given an agent with corresponding checkpoints, we define the individual predicted makespan of the agent to be the minimum number of steps required to visit all checkpoints (without the need to avoid collisions with other agents).
3.1.5 Relative predicted makespan

Our motivation for measuring relative predicted makespan is two-fold. First off, relative predicted makespan gives an indication of what trends in makespan difference are apparent. For instance, high relative predicted makespan error can indicate a lot of conflicts. Moreover, as it is used in the heuristics, it might be beneficial to see how accurate the heuristics 'guesses' are. In applications in real-time, measuring relative prediction error is beneficial for planning.

3.1.6 Global predicted makespan

Given a set of agents, each with corresponding checkpoints, we define the global predicted makespan to be the maximum of all individual predicted makespans of the agents. Although this is just an approximation of the true makespan, it is much faster to compute.

3.1.7 Makespan prediction error

Given a set of agents, each with corresponding checkpoints, we define the makespan prediction error to be the absolute difference between the global predicted makespan and the true makespan (after avoiding collisions) of an optimal set of paths that follow the checkpoints. The true makespan is thus always greater than or equal to the predicted makespan.

3.1.8 Relative makespan prediction error

The relative makespan prediction error for a set of agents, each with corresponding checkpoints, is defined as the makespan prediction error divided by the true makespan of an optimal set of paths that follow the checkpoints.

3.2 Solving MAITP

The algorithm we propose in this study is a two-phase algorithm where the first phase is a heuristic for assigning checkpoints to agents and the second phase uses conflict-based search and an A*-based algorithm to find optimal paths given these checkpoints. This means that the algorithm as a whole is not necessarily optimal, and that the heuristic used has an impact on the quality of the solution. Although not proven, we expect the multi-agent item transportation problem to be NP-hard, which is why a heuristic is proposed instead of an optimal algorithm.
In order to guarantee optimal paths for the task of visiting the checkpoints, the regular A* algorithm was modified to take into account an ordered list of destination nodes and a progress counter for each path that holds the number of already visited checkpoints. Because all agents need to have equally long paths (see the MAITP definition), the algorithm was also extended with a parameter denoting the minimum number of steps needed to travel while visiting the checkpoints (or after reaching the last checkpoint). If the path is too short when reaching the last checkpoint, the search is changed to a breadth-first search, without a destination, with the sole purpose of extend the path. If the returned path is longer than the minimum number of steps required, the paths for all other agents need to be extended with this new length as the minimal number of steps (and this continues until all paths are equally long). If at any point this modified A* algorithm fails to find a long enough path leading through all checkpoints, the whole algorithm fails at finding a solution to the problem. The algorithm could potentially be modified to try with other checkpoints but we made the assessment that this was unnecessary given the scope of the study.

In this study we investigated a few choices of heuristic for the first phase of the algorithm, where the tested heuristics were evaluated based on the final makespan after having avoided all conflicts during the second phase.

### 3.3 Heuristics used

All heuristics used in this study were simple cost-based heuristics that used only a few metrics. There were two main reasons behind this choice:

1. Having all heuristics in the same category enables highly interpretable comparisons between them as the variability between them is minimized,

2. We wanted especially to examine what environment-specific patterns are better than others for making choices, and we assessed that this could be done by using environment-specific metrics for otherwise simple heuristics.

In total there were 11 heuristics used in the study, each with the basic structure of iteratively adding a checkpoint to an agent until all items have been visited and dropped off. In every iteration, all agent-item and agent-drop-off pairs are assessed using a cost function, after which the pair with the lowest cost is chosen. The agent in the chosen pair is then assigned the position of the item or drop-off as its next checkpoint. When any agent’s inventory is full, the agent is assigned to visit the drop-off with the minimum cost. When all items have
been visited, every agent is assigned to go to its corresponding minimum-cost drop-off.

Atop the basic structure described above, a base cost was included in each of the 11 cost functions. This base cost (equal for all 11 functions), given an agent and position, is defined as the individual predicted makespan for the agent (for all currently determined checkpoints) added to the graph distance from the agent’s last determined position to the specified position. The purpose of having a base cost is to provide enough of a foundation to be able to solve the test cases deemed necessary for the study without including too complex decision rules that decrease interpretability of the results. This base cost was assessed to be appropriate in these regards as it was barely able to solve the chosen test instances while being as simple as we could come up with.

What makes the heuristics different from one another is the addition of 5 environment-specific metrics. Each metric, added to the base cost, constitute one complete cost function (making 5 different such cost functions). In addition, each metric with inverted sign is added to the base cost, making another 5 cost functions. For the last cost function, only the base cost is used. This is to provide a baseline to allow us to measure how adding the metrics affected the performance of the heuristics.

The metrics are defined as follows, each with regards to a specified agent (having at least the start vertex as an already determined checkpoint together with a remaining carrying capacity) and a position corresponding to either a drop-off or an item:

(1) The average distance from the last checkpoint to all other items. The idea behind this cost is to get the agents to tend towards the middle of the items and thus have many possibilities of choosing the next item. When the sign of this cost is inverted, the hypothesis is instead that the agents tend to begin with the most isolated items and working inwards.

(2) The average distance from the last checkpoint to the closest items, where the number of considered items are equal to the remaining capacity of the agent. The idea behind this approach is similar to metric (1) but only planning one trip ahead in hopes of getting a reading more relevant to the agents’ current situations.

(3) The average distance from checkpoint to checkpoint when travelling in a sequence starting at the last determined checkpoint for the agent and repeatedly choosing the closest item until the inventory is full. The idea behind this cost is to incorporate the tendency of picking items nearby
(from the base cost) to predict the sequence of items picked until drop-off.

(4) The average distance from the last determined checkpoint to all agents. The idea behind this metric is to provide a way for the agents to spread out and avoid each other. When the sign is inverted, the agents are thought to instead try to be close to each other.

(5) The distance from the last determined checkpoint to the closest drop-off, divided by the agent’s remaining capacity. The idea here is to make the agents more likely to travel closer to drop-offs when their inventories are almost full and soon need to drop off their items.

3.4 Experiments

The purpose of the experiments was to examine how the heuristics compare to the base heuristic and to each other under varying problem instance characteristics and how the quality of the solutions depend on the characteristics. To study this, we chose to test the heuristics on instances that all have 100 nodes but varying values on the following 5 parameters:

- The number of edges (with possible values 300, 400, 500, 600),
- The number of drop-off points (with possible values 4, 6, 8, 10),
- The number of items (with possible values 20, 30, 40, 50),
- The number of agents (with possible values 1, 2, 3, 4),
- The capacity of the agents (with possible values 4, 6, 8, 10).

The reason for keeping the number of nodes fixed is to allow for some of the other parameters to vary the actual density in the graph instead of everything just scaling. We wanted to have as many different parameter values for each parameter as possible to be able to find trends in the connection between parameters and solution quality but found also that having many more than 4 values each took too long to compute. We also wanted to have the parameter values somewhat reflect possible real-world applications such as robotic warehouses, autonomous taxi services, etc. The problem was that decreasing the number of edges and drop-off points to a seemingly reasonable number made the instances harder to solve. This was also the case when increasing
the number of agents and the number of items, and decreasing agent capacity. We made the judgment that these parameter values constitute a reasonable balance.

As mentioned in the scope all item weights were set to 1 and all agent capacities were homogeneous. This was done to avoid getting overwhelmed with parameters to vary and take into consideration, in order to get more easily interpretable results.

In total 22 datasets were used for testing, each containing one instance for every unique combination of parameter values, giving $4^5 = 1024$ problem instances in each dataset. All instances were randomly generated although with constraints to follow the specified parameter values and to comprise a connected graph (otherwise the instance would be unsolvable if for example there exist items on a component that does not contain a single drop-off point).

Each heuristic was tested on each problem instance in each dataset, and the makespan together with the relative makespan prediction error were recorded for each such solution. In cases where at least one heuristic was unable to solve an instance within a dataset, the entire dataset was deleted and regenerated using another random seed, after which all heuristics were retested on that dataset and the solutions to the previous dataset disregarded. This was to ensure that all heuristics were tested on the exact same instances and with the same number of data points. The reason for failure was always that the optimal search took too long, so there was no way of knowing whether the heuristic generated unsolvable checkpoints or just would have taken very long to find optimal paths.
Chapter 4

Results

4.1 Cost metrics

The cost metrics ‘All items’ and ‘Closest drop-off’ seemed to be an improvement over the base cost. Even if all metrics are based on the base cost, these two seemed to outperform it when negative metrics were applied (see the baseline in figure 4.1).

An interesting finding with regards to figure 4.2 was that ‘All agents’ improved significantly when using a negative metric rather than a positive.
Figure 4.1: For each cost metric (positive and negative), the average makespan within each of the 22 datasets (over the 1024 instances therein) were calculated. The sample means of these 22 averages for each cost metric are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom. The horizontal black line shows the corresponding sample mean for the base cost, with the gray lines forming an approximate 95% confidence interval.
Figure 4.2: For each cost metric (positive and negative), the average relative makespan prediction error within each of the 22 datasets (over the 1024 instances therein) were calculated. The sample means of these 22 averages for each cost metric are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom. The horizontal black line shows the corresponding sample mean for the base cost, with the gray lines forming an approximate 95% confidence interval.

### 4.2 Number of agents

One could quite clearly see a relationship between an increase in the number of agents and a decrease in the average makespan (see figure 4.3). Moreover, there was also a clear increase in relative makespan prediction error when the number of agents increased (see figure 4.4).

What stood out in figure 4.4 was the huge discrepancy between 'All agents' negative and the rest.
Figure 4.3: For each negative cost metric and the base cost, the average makespan over the instances corresponding to each tested number of agents (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of agents are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.
Figure 4.4: For each negative cost metric and the base cost, the average makespan prediction error over the instances corresponding to each tested number of agents (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of agents are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.

4.3 Agent carrying capacity

One could quite clearly see a relationship between an increase in agent carrying capacity and a decrease in makespan.

The makespan prediction errors were so varying (see figure 4.6) for all heuristics except ’All agents (negative)’. 
Figure 4.5: For each negative cost metric and the base cost, the average makespan over the instances corresponding to each tested capacity (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested capacity are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.
4.4 Drop-offs

Although a general trend can be seen where an increased amount of drop-offs both decreases makespan and prediction error. There is a lot of inaccuracy as it pertains to these results (see 4.7) as the confidence intervals where quite broad. Yet again, it seems ‘All agents (negative)’ had quite a low prediction error.
Figure 4.7: For each negative cost metric and the base cost, the average makespan over the instances corresponding to each tested number of drop-off points (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of drop-off points are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.
4.5 Edges

Similar to the previous chapter, a general trend is seen where as edges increase, the average makespan decreases (figure 4.9). However, as the number of edges increases, prediction error seems to do so too. Yet again, the results are highly variable and so no certain conclusions can be drawn but there seem to be indications.
Figure 4.9: For each negative cost metric and the base cost, the average makespan over the instances corresponding to each tested number of edges (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of edges are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.
Figure 4.10: For each negative cost metric and the base cost, the average makespan prediction error over the instances corresponding to each tested number of edges (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of edges are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.

4.6 Items

As the number of items increases, average makespan does too (figure 4.11). The results for prediction error are inconclusive (figure 4.12) except for 'All agents (negative)' that yet again outperformed all other heuristics.
Figure 4.11: For each negative cost metric and the base cost, the average makespan over the instances corresponding to each tested number of items (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of items are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.
Figure 4.12: For each negative cost metric and the base cost, the average makespan prediction error over the instances corresponding to each tested number of items (256 such instances), within each of the 22 datasets, were calculated. The sample means of these 22 averages for each such metric and tested number of items are shown on the y-axis, together with an approximate 95% confidence interval of each such sample mean. The confidence intervals were calculated using the t-distribution with 21 degrees of freedom.
Chapter 5

Discussion

In general we could see that 'All items (negative)' and 'Closest drop-off (negative)' seemed to be metrics that outperformed the base cost in terms of makespan. The confidence interval of the base cost slightly overlaps with that of 'Closest drop-off (negative)', and so this one is not quite as certain. One might think that all items being spread out, with a large average distance between them reduced collisions to a large extent, however figure 4.2 in fact the number average relative makespan prediction error was higher than that of the base cost. As for the rest of the cost metrics, one cannot say if they’re better than the base cost but it is possible. We cannot discard these metrics as there might be unexplored cases, or other uses in which these metrics as better than the base cost, for example adding weights to certain metrics.

The weighting of different metrics might have been inappropriate and thus the true potential might not have been discovered. As seen in a few of the figures, the confidence intervals were quite large and thus it is hard to point to any concrete trend. We did however, consistently, see that the prediction error was significantly lower for 'All agents (negative)'.

The results indicate trends such as makespan decreasing as a result of increases in number of agents, carrying capacity, number of drop-offs, and edges. An increase in agents might mean more items per unit time can be picked up and so makespan decreases, this is only up to a certain point however, as too large a number of agents on a small graph can induce many collisions or cause some of the agents to be idle. Carrying capacity allows agents to have to visit drop-offs less frequently, and thus reduced flocking around these points, and of course the amount of trips back and forth. Subsequently this decreases makespan as well. As the number of drop-offs increases, agents have more places to drop their items and it seems that more is always better. Lastly,
one could hypothesize that as edges increase so does the potential to avoid conflicts, and thus reduces makespan. However, figure 4.10 could not show such a trend, but instead a slight increase of the average relative makespan prediction error as the number of edges increased.

As for prediction errors, ’All agents (negative)’ was designed to minimize this and it clearly does outperform the other heuristics. One might suppose that this is due to that agents flow better and avoid collisions to a higher extent. Furthermore, we saw for instance that ’All agents (positive)’ performed relatively poor in both figure 4.1 and figure 4.2, and this is a possible indication that excessive collisions are indeed costly with regards to minimizing makespan. Although it is somewhat of an indicator, not all increase in makespan might necessarily be attributable to collisions. For instance, it could be that items are ’stolen’ by agents in proximity to each other.

There are a number of flaws in this study, for example the very limited search for appropriate weights or uses of the metrics as well as a limited number of test instances that in many cases did not provide small enough confidence intervals for statistical significance. Our chosen parameters limit the scope of the study and the applicability as, for instance, we only carried a very small amount if items. Capacities, number of agents, graph-size, and drop-offs are also parameters that might have had a significant effect. In addition to this, the limitations in terms of item weights set to 1 and homogeneous carrying capacities probably also affected how well these results reflect real problem scenarios.

Future research of the problem might include determining whether the MAITP is NP-hard or not and as well as approximability bounds. As for the heuristics, future extensions might include combining several different cost metrics and optimizing their weights. Another possible area to explore is comparing different heuristics to optimal solutions to explore just how much efficiency there is to reap in bettering the algorithms. This was outside the scope of our study and thus was excluded.
Chapter 6

Conclusion

This study has shown two potentially useful metrics, 'All items (negative)' and 'Closest drop-off (negative)', that have outperformed the simple base cost heuristic and show promise in approximately solving the MAITP problem. Moreover, factors such as carrying capacity, number of drop-offs, number of edges, and number of agents all were of significance regarding makespan, with negative correlation. In contrast, the number of items seemed to have a positive correlation with the makespan. Furthermore, one can see indications of dispersed agents decreasing relative prediction error as compared to the base cost.
Bibliography


