Reducing blocking effects in multi-block layouts

Felix Werth, Oliver Ullrich, Ewald Speckenmeyer
{werth|ullrich|esp}@informatik.uni-koeln.de
Institut für Informatik, Universität zu Köln
Pohligstraße 1, 50969 Köln, Germany

Abstract

Tour planning in multi-block layouts is a common exercise in logistics. In those systems, blocking effects result from conflicting agents competing for resources. Although clearly exceptional in real world applications, most methods of tour planning assume only one active agent, and thus do not consider blocking effects.

In this paper we examine heuristic methods of tour planning in multi-block layouts with multiple agents, finding that blocking effects have a significant impact on system performance. We show that methods devised for the mentioned special case do not scale very well when applied to scenarios with multiple agents. We propose a heuristic method which is capable of reducing blocking effects. It generates tours of equal or shorter length than those produced by the other examined methods.

1   Introduction

The layout of many real world systems can be considered as a multi-block configuration. Examples include warehouse floors, parking lots, or supermarkets. Tour planning is a common task related to those systems.

Most methods of tour planning assume generated tours to be independent, and take it for granted that they do not interfere, even if executed concurrently. This assumption is not self-evident, considering that in most real world problems there are lots of active workers or vehicles, which we call agents for our concerns. As shown in [12], blocking effects occur in those systems and can have a significant impact on system performance.

In this paper we examine to what degree blocking effects in multi-block layouts downgrade system performance, and how this impact can be reduced. We define blocking effects as the relative increase in average tour duration caused by deploying multiple agents simultaneously. This can easily be converted into other performance indicators like order throughput.

We begin by describing the background of blocking effects in multi-block layouts with multiple agents and discuss related work (Section 2). We then show that commonly applied heuristics do not scale very well under the mentioned conditions, especially in configurations with high utilization, and propose a method optimized for the described multi-block layout with multiple active agents (Section 3). A line of experiments is then conducted on those methods, whose results are described and discussed (Section 4). We close with a short summary of the lessons learned and some thoughts on further research (Section 5).

2 Background

2.1 Blocking effects in multi-block layouts

A common multi-block layout (e.g. found in warehouses) consists of access points along parallel aisles, which are subdivided by cross aisles. (see fig. 3a) Aiming for high space utilization during layout planning often results in narrow aisles. This leads to situations where two or more agents interfere with each other. The effect is a decrease in overall system performance, since agents are slowed down or blocked. We regard three types of blocking in this paper: Concurrent access to a storage location, moving passed occupied access locations and head-on collisions in aisles. (see fig. 1) In situations of the first type, an agent is completely blocked when trying to access the next location on its tour. It has to wait until the access location is freed. Agents engaged in situations of type two and three have to draw aside and thus are slowed down on their travel.
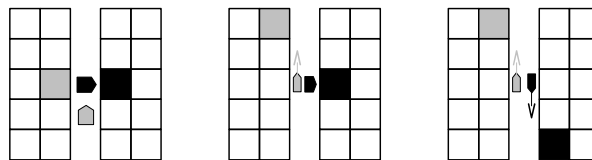

Figure 1: Three types of blocking situations

The blocking effects' impact depends on several aspects of the system under consideration, like layout size and number of agents deployed. Routing methods and storage assignment strategies further influence system behavior.

Most often blocking effects are neglected and a linear relationship between number of agents and key performance indicators is assumed. We call this the static case. In the dynamic case, blocking effects are no longer ignored and their effect is explicitly taken into account.

For the rest of this paper, we adopt part of the terminology of order picking for simplicity's sake. In this context, items to be fetched or stored at an access point are called picks.

2.2 Related work

In general it is not possible to find an optimal solution to the routing problem in feasible time since it is known to be NP-hard. A well known method to compute an optimal solution on a general layout using exponential time in the number of items to be picked was introduced by Held and Karp. [5] This is a major improvement compared to the brute force combinatorial method having factorial time complexity but it is still not applicable to large number of picks. Because of this a lot of heuristic methods have been developed for several applications.

Concerning the domain of order picking the static case has been extensively studied. Ratliff and Rosenthal developed a linear time algorithm for calculating optimal solutions in 1-block layouts. [9] Their result was extended to series-parallel graphs by Cornuéjols et al. [1] Roodbergen and de Koster applied the same technique to 2-block layouts. [11] Distance approximations of some common heuristics were presented by Hall. [4] Optimal and heuristic solutions were compared by de Koster et al. [6]

The dynamic case has received far less attention. Some authors dealt with measuring blocking effects under different system configurations. Existing heuristics developed for the static case were used without exceptions. Gue et al. built an analytical model to quantify congestion for the *s-shape* heuristic in 1-block layouts, assuming deterministic pick times. [3] Their work was extended to non-deterministic pick times by Parikh and Meller. [8] Furmans et al. demonstrated how to transform a manual order picking system into a queueing model. After comparing the analytical results obtained from queueing theory to simulation results they concluded that queueing theory is a suitable means of modeling manual order picking systems. Their results showed that blocking can reduce overall throughput by 10% in typical implementations. [2]

3   Heuristic approach

3.1   Common heuristics for tour planning in multi-block layouts

One can distinguish general purpose heuristics which work on arbitrary layouts from special case heuristics which take the layout's structure into account. A common heuristic of the former kind is *nearest neighbour,* prominent examples of the latter for block layouts with parallel aisles are *s-shape*, *largest gap* and *return*. All these heuristics were developed for the static case, minimizing the length of a single tour. No interdependencies between parallel tours were regarded, and thus no blocking effects have been considered.

S-shape constructs tours blockwise starting with the block farthest away from the depot. For each block every aisle containing picks is entirely traversed with alternating directions. Subaisles are visited from left to right or vice versa, skipping empty subaisles. (see fig. 2a)

Largest gap applies a blockwise scheme similar to *s-shape* but enters every aisle containing picks only up to the largest gap between two adjacent picks, taking exception for the leftmost or rightmost relevant aisle, which is traversed entirely. (see fig. 2b)

Nearest neighbour generated tours start from the depot and go on to the nearest location which has not been visited yet. They continue iteratively to the nearest unvisited location until all locations have been visited and return to the depot. (see fig. 2c)

For single block layouts it was shown in [12] that a *return* strategy is outperformed by all other described strategies. For this reason *return* is not paid further attention to in the remainder of this paper.

## 3.2   A blocking avoidance heuristic

To yield good results with rising system utilization, a tour planning method has to adhere to two conditions: First, the generated tours must be short, and second, the tour planning has to avoid collisions and their associated costs.

To generate short tours a method has to exploit the given grid-like layout structure with comparatively short pick aisles and connecting cross aisles to avoid detours.

To avoid collisions, tours should adhere to the following conditions: First, aisle space should be utilized evenly. Uneven utilization leads to congestion in parts of the system, resulting in bottlenecks and high numbers of collisions. Second, cross aisles should be traversed unidirectionally. Conflicting agents in cross aisles are a major source of collision costs, which can be avoided by moving through the cross aisles in a common
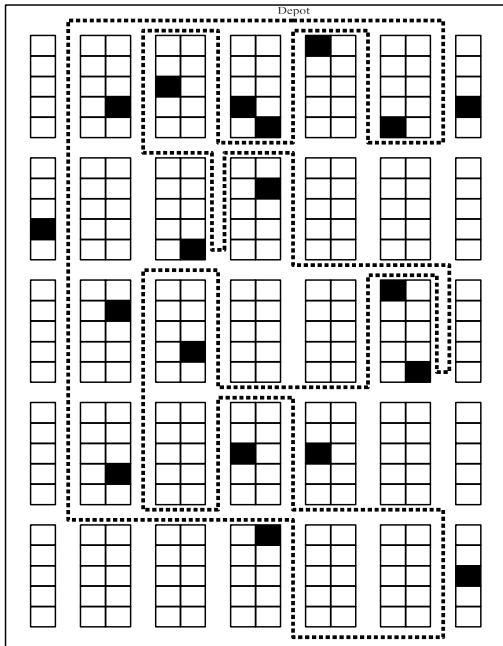
a) S-shape
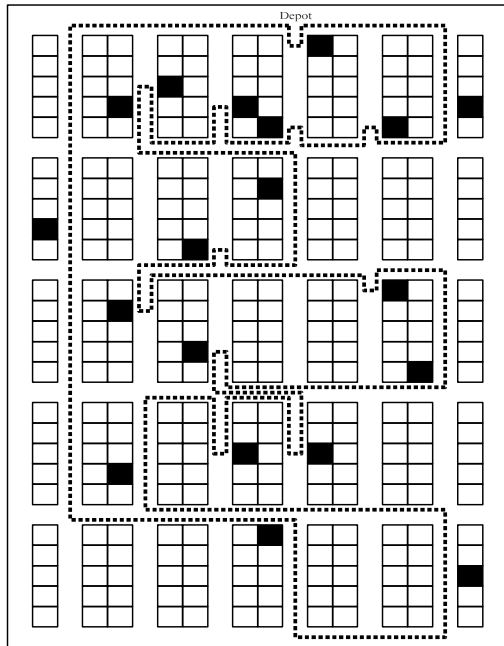
b) Largest gap
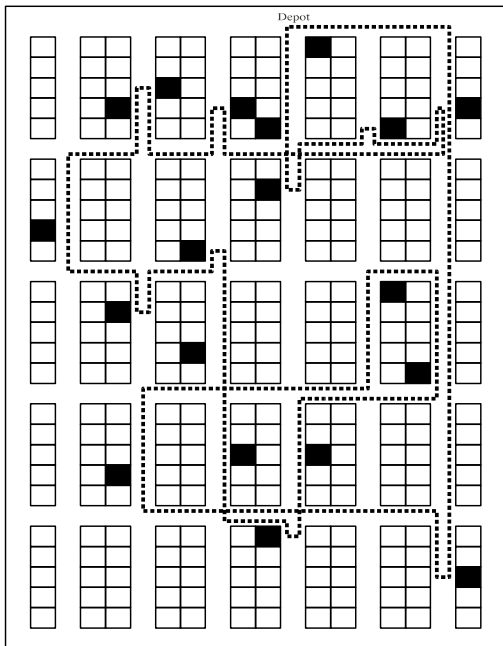
c) Nearest neighbour

d) WUSH

Figure 2: Example tours for four heuristics

a) Terminology

b) Subpath construction

Figure 3: Terminology and subpath construction

direction. Congestion of aisles caused by agents picking cannot be avoided. It is possible though to minimize the time agents traverse through pick aisles and are thus subjected to potential blocking.
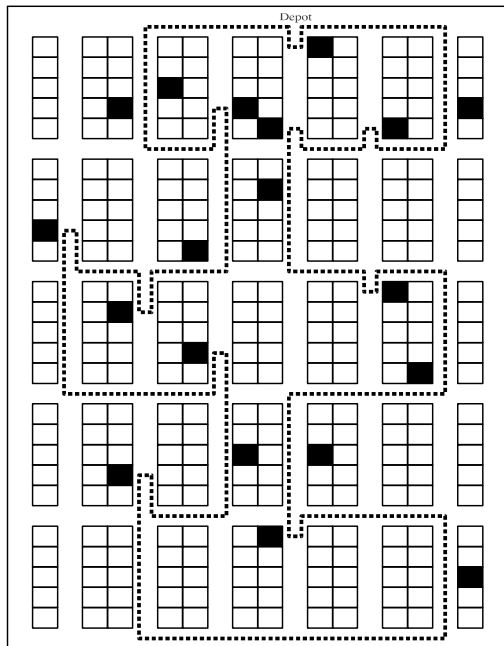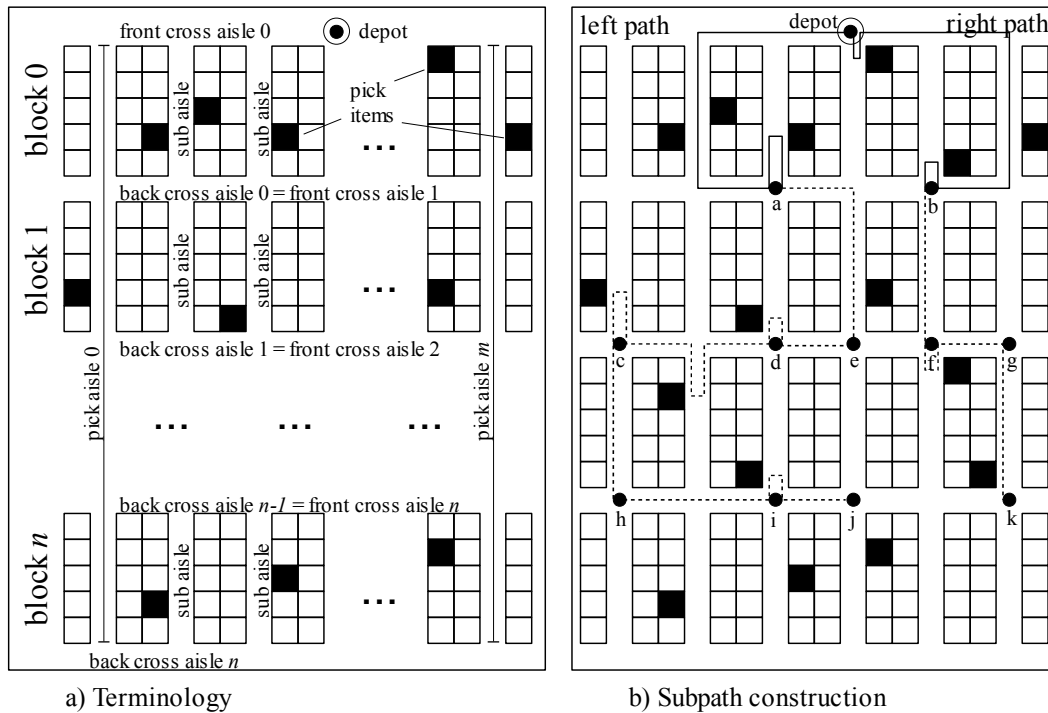
Following this strategy we impose an alternating one-way traffic scheme on cross aisles starting with leftward traffic on the first cross aisle where the depot is located.

Algorithm WUSH:

1. **for each** cross aisle c **do**
   for each subaisle s **do**
      assign all picks in s from adjacent blocks to c regarding largest gap
2. **for each** second block b **do**
   i. determine the best combination of pick aisles for entering front cross aisle and exiting back cross aisle of b on the left path
   ii. add left subpath to left path starting from last position on left path
   iii. add right subpath to right path starting from last position on right path
3. assemble path p by combining left and right path

Figure 4: The proposed method

A tour is constructed blockwise. We generate a left path and a right path which are combined on the first front cross aisle and on the last back cross aisle. (see fig. 2d) The left path corresponds to the way down from the depot to the farthest cross aisle, while the right path corresponds to the way up. The method consists of three steps. (see fig. 4)

In a preprocessing step, we apply a *largest gap* strategy to each subaisle to map every item either to the front cross aisle or to the back cross aisle of the corresponding block. In

a second step, we iterate over every second block to construct the left path and the right path. Finally the two paths are combined into the resulting tour.

Algorithm 2.ii. Construct left subpath:
1. collect all picks on the following path:
2.   go right from last position on previous back cross aisle to best entry pick aisle
3.   go down to front cross aisle
4.   go left to the leftmost pick aisle of front and back cross aisle
5.   go down to back cross aisle
6.   go right to best exit cross aisle
7.   left ending pick aisle = best exit pick aisle

Figure 5: Constructing left sub path

Main part of the algorithm is planning the left and right subpaths for a block *i.* (see fig. 5 and 6) The previous iteration for block *i-2* resulted in a left path ending (see fig. 3b point *a*) and a right path ending (point *b*) accordingly. During the current iteration, care has to be taken of those items, which have to be picked from both front and back cross aisle of block *i*. This leads to the decision which items should be picked on the left path leaving the other items for the right path. Part of the decision is made by choosing a pick aisle to enter the front cross aisle of block *i* from a set of potential entry pick aisles (points *d, e*). Each item to the right of that pick aisle has to be picked on the way back. On the front cross aisle agents are only allowed to move left due to the one-way restriction. The left subpath is extended to the leftmost relevant pick aisle (point *c*). This pick aisle is used to go down to the back cross aisle of block *i*. Now the pick aisle to exit the back cross aisle of block *i* has to be chosen, which leaves any unpicked items on the back cross aisle to be picked on the right subpath. After having decided where to exit the back cross aisle from a set of potential exit pick aisles (points *h, i, j, k*), the right subpath is constructed by collecting any unpicked items from front and back cross aisle of block *i,* thereby regarding traffic directions.

Algorithm 2.iii. Construct right subpath:
1. collect all picks on the following path:
2.   **if** first relevant pick aisle *f* right of best entry pick aisle on front cross aisle is left of
                        right ending pick aisle **then**
                                go left to *f* on previous back cross aisle
3.   go down to front cross aisle
4.   go right to rightmost pick aisle of front and back cross aisle
5.   go down to back cross aisle
6.   go left to first pick aisle p with picks right to best exit cross aisle
7.   right ending pick aisle = p

Figure 6: Constructing right sub path

The decision where to enter the front cross aisle and where to exit the back cross aisle of block *i* is made by a greedy approach. For every combination of potential entry and exit pick aisle the length of both left and right subtour is computed. Finally the combination which yields the minimum subtour length is chosen. (see fig. 7)

Lacking a better name, we modestly call the described heuristic after ourselves, which makes it *wush*.

Algorithm 2.i. Determine best combination **for** block *b*:

1. set of potential entry pick aisles = { left ending pick aisle }
2. **for each** pick aisle *p* with picks on front cross aisle of *b* **do**
   if *p* is between left ending pick aisle and right ending pick aisle **then**
     potential entry pick aisles = potential entry pick aisles ∪ *{p}*
3. set of potential exit pick aisles = { leftmost pick aisle on front cross aisle }
4. **for each** pick aisle *p* with picks on back cross aisle **do**
   potential exit pick aisles = potential exit pick aisles ∪ *{p}*
5. length of best subtour = ∞
6. **for each** potential entry pick aisle *p1* **do**
  for each potential exit pick aisle *p2* **do**
    calculate length of left and right subtour for *p1* and *p2*
      if length of current subtour < length of best subtour **then**
       length of best subtour = length of current subtour
       best entry pick aisle = *p1*
       best exit pick aisle = *p2*

Figure 7: Determine best combination of entry and exit aisles

4   Experiments

4.1   Modeling blocking effects in multi-block layouts

To evaluate the proposed method, we built an agent based software application. Agent based modeling and simulation is especially useful for examining global dynamics of a system whose rules of behaviour are set on a local or individual level. [7] In our model, while tours are assigned locally to individual agents, we want to observe the resulting system's global key data, especially the average tour duration.

An agent based model consists of the agents themselves (with their set of attributes and set of behavioral rules), their methods of interaction and the environment in which they move and interact with. [7]

In our application, the environment is modeled as an undirected, weighted graph. Nodes represent picking points and junctions of aisles. Edges represent aisles between those points, weighted by their length *l*.

Agents move along the edges with speed *v*, following their assigned tour. If agents collide on an edge, they adjust their speed to v_{coll}. When picking, an agent blocks a node, unblocking it after t_{block}. Agents have no knowledge about other agents' tours, so they cannot avoid collisions by adaptive tour planning. When an agent is done with its tour, it gets assigned the next one from a global queue. If there are no more tours to execute, the agent stops.

We compare the proposed heuristic to *largest gap*, *s-shape* and *nearest neighbour*, and also include *random* for good measure. The agents' speed is assigned to *v=120cm/sec*. In case of collision the agents slow down to v_{coll}*=24cm/sec*. The length of edges is set to 100cm for pick aisles and 200cm for cross aisles, reflecting the distance between two pick points and two pick aisle entrances, respectively. Access time t_{block} is set to 10sec.

We build scenarios by varying the number of agents (1 to 25), the number of items per tour (10 to 25) and two layout sizes (5 and 7 blocks). For each of those scenarios we execute 30 simulation runs with 1,000 picking tours each.

4.2 Results and discussion

To evaluate the absolute performance the average tour duration has to be taken into account. Regarding the static case, the proposed strategy dominates *random*, *largest gap*, and *s-shape* at every point of measurement (see table 1, col. 5). In the smaller configuration of 5 blocks, *nearest neighbour* yields a slight advantage if tours have 10 to 20 picks (2.4% at 10, 1.0% at 15, 0.01% at 20 picks), but at 25 picks *wush* generated tours save a modest 0.9% of time. In 7-block configurations the proposed heuristic dominates all contestants.

Method	No. of blocks	No. of items	Average tour length	Ø tour duration (static case)	Number of agents (dynamic case)				
					5	10	15	20	25
Random	5	10	21391	178258	3,08%	6,92%	10,78%	14,56%	18,30%
	5	15	30788	256567	3,14%	7,09%	10,98%	14,82%	18,64%
	5	20	40109	334242	3,19%	7,18%	11,12%	15,04%	18,93%
	5	25	49472	412267	3,21%	7,23%	11,22%	15,14%	18,67%
	7	10	26889	224075	2,13%	4,78%	7,47%	10,05%	12,61%
	7	15	38496	320800	2,15%	4,89%	7,57%	10,21%	12,85%
	7	20	50134	417783	2,20%	4,92%	8,36%	10,31%	12,97%
	7	25	61719	514325	2,22%	4,95%	8,07%	10,35%	13,03%
Largest Gap	5	10	13899	115825	2,64%	6,09%	9,27%	12,55%	15,85%
	5	15	17605	146708	2,81%	6,35%	10,06%	13,24%	16,65%
	5	20	20818	173483	2,98%	6,90%	10,51%	13,91%	17,47%
	5	25	23625	196875	2,94%	6,81%	10,79%	14,55%	17,98%
	7	10	16316	135967	1,86%	4,19%	6,44%	8,64%	10,89%
	7	15	20244	168700	2,03%	4,61%	6,98%	9,40%	11,67%
	7	20	23808	198400	2,16%	4,86%	7,47%	9,95%	12,49%
	7	25	27051	225425	2,11%	5,02%	7,65%	10,34%	12,89%
S-Shape	5	10	14126	117717	2,69%	6,06%	9,35%	12,65%	15,87%
	5	15	17933	149442	2,86%	6,51%	10,01%	13,29%	16,53%
	5	20	21165	176375	3,00%	6,83%	10,39%	13,85%	17,50%
	5	25	23908	199233	2,98%	6,77%	10,90%	14,77%	18,13%
	7	10	16521	137675	1,85%	4,22%	6,51%	8,64%	10,82%
	7	15	20592	171600	2,02%	4,56%	6,87%	9,37%	11,44%
	7	20	24239	201992	2,12%	4,82%	7,31%	9,81%	12,17%
	7	25	27502	229183	2,22%	4,95%	7,60%	10,14%	12,86%
Nearest neighbour	5	10	12640	**105333**	3,14%	7,10%	10,74%	14,41%	18,10%
	5	15	15384	**128200**	3,27%	7,28%	11,28%	15,17%	18,96%
	5	20	17874	**148950**	3,39%	7,59%	11,75%	15,81%	19,67%
	5	25	20139	167825	3,51%	7,95%	12,19%	16,49%	20,44%
	7	10	15514	129283	2,22%	4,95%	7,56%	10,19%	12,82%
	7	15	18460	153833	2,31%	5,20%	7,97%	10,79%	13,32%
	7	20	21123	176025	2,37%	5,38%	8,28%	11,14%	13,85%
	7	25	23642	197017	2,46%	5,55%	8,67%	11,51%	14,57%
wush	5	10	12943	107858	**2,21%**	**5,01%**	**7,68%**	**10,20%**	**12,79%**
	5	15	15553	129608	**2,33%**	**5,47%**	**8,33%**	**11,02%**	**13,81%**
	5	20	17888	149067	**2,60%**	**5,98%**	**9,15%**	**11,87%**	**15,04%**
	5	25	19968	**166400**	**2,90%**	**6,38%**	**9,90%**	**13,09%**	**16,23%**
	7	10	15456	**128800**	**1,68%**	**3,76%**	**5,69%**	**7,46%**	**9,46%**
	7	15	18175	**151458**	**1,71%**	**3,96%**	**5,96%**	**7,84%**	**9,73%**
	7	20	20715	**172625**	**1,87%**	**4,24%**	**6,47%**	**8,45%**	**10,40%**
	7	25	23086	**192383**	**2,01%**	**4,51%**	**6,96%**	**9,03%**	**11,29%**

Table 1: Blocking effect for five heuristics

Concerning the blocking effect of the dynamic case, the proposed strategy dominates all other methods. Independent of layout, order size and number of agents deployed *wush* generated tours have the lowest increase in average tour duration. This means that if the proposed method dominates another method in the static case it remains dominant in terms of absolute tour duration in the dynamic case. Three configurations of the static case remain where *wush* is outperformed by *nearest neighbour*. Those are examined further to find the break even point where the reduction of blocking effects on the part of the proposed method outweighs the slightly lower average tour duration achieved by *nearest neighbour*. In the 5-block configuration, the tours generated by *nearest neighbour* produce better results, if both the number of picks per tour and the number of agents are small (see table 2). At all other combinations of parameters, the proposed heuristic yields the shortest tour durations.

Method	No. of blocks	No. of items	Average tour length	Ø tour duration (static case)	Number of agents (dynamic case)				
					5	10	15	20	25
Nearest	5	10	12640	**105333**	**108637**	**112808**	116646	120511	124397
neighbour	5	15	15384	**128200**	**132391**	137529	142664	147646	152501
	5	20	17874	**148950**	153997	160252	166448	172496	178255
	5	25	20139	167825	173724	181172	188282	195503	202130
wush	5	10	12943	107858	110242	113262	**116141**	**118865**	**121654**
	5	15	15553	129608	132629	**136698**	**140410**	**143889**	**147509**
	5	20	17888	149067	**152939**	**157976**	**162711**	**166767**	**171490**
	5	25	19968	**166400**	**171226**	**177022**	**182866**	**188188**	**193415**

Table 2: Comparison of average tour duration of nearest neighbour and wush

The proposed method is optimized for multi-block layouts with multiple active agents. Even with just one active agent *wush* competes well since it generates tours of comparatively short length, and thus dominates most of the more general routing methods like *s-shape*, *largest gap* and *random*, which yield considerably longer tours in our layout. The remaining competitor is the *nearest neighbour* method, which produces tours of comparable length.

When expanding to multiple agents, *s-shape*, *random*, and *largest gap* do not avoid conflicts and thus yield relatively high collision costs. Additionally, their generated tours are comparatively long, so that they perform even worse when applied to a configuration with high utilization. *Nearest neighbour*, on the other hand, produces even more collisions because concurring tours form a unpredictable pattern with no regard to collisions.

5 Conclusion and future work

In this paper we examined heuristic methods for tour planning in multi-block layouts with multiple agents. We found that blocking effects have a measurable impact on system performance, especially in configurations with high utilization. We showed that methods devised for the special case of one active agent, like *s-shape*, *nearest neighbour*, or *largest gap*, do not scale well when applied to scenarios with more active agents.

The proposed heuristic is capable of reducing blocking effects considerably in every instance of the dynamic case. Considering tour lengths, it competes well against *nearest neighbour* and generates significantly shorter tours than all other examined methods.

After working on the heuristic approach, it seems natural to look for an exact solution. It might be possible to find such an algorithm feasible for configuration sizes found in real world instances. A first step would be to simplify the problem radically without losing its core characteristics.

Even if it should not be feasible to employ such an exact method for runtime reasons, insights won by examining a simplified problem might spawn better heuristic solutions.

6   References

[1]   *Cornuéjols, G., Fonlupt, J., Naddef, D.*: The traveling salesman problem on a graph and some related integer polyhedra. Mathematical Programming 33 (1985), No. 1, pp. 1-27.

[2]   *Furmans, K., Huber, C., Wisser, J.*: Queueing Models for manual order picking systems with blocking. Logistics Journal 01 (2009), http://www.logistics-journal.de/archive/2009/2092.

[3]   *Gue, K. R., Meller, R. D., Skufca, J. D.*: The effects of pick density on order picking areas with narrow aisles. IIE Transactions 38 (2006), pp. 859-868.

[4]   *Hall, R. W.*: Distance approximations for routing manual order pickers in a warehouse. IIE Transactions Vol. 30 (1985), No. 1, pp. 76-87.

[5]   *Held, M., Karp, R. M.:* A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics 10 (1962), No. 1, pp. 196-210.

[6]   *de Koster, R., van der Poort, E.*: Routing orderpickers in a warehouse: a comparison between optimal and heuristic solutions. IIE Transactions 30 (1998), pp. 469-480.

[7]   *Macal, C. M.; North, M. J.*: Tutorial on agent-based modelling and simulation. Journal of Simulation 4 (2010), pp. 151-162.

[8]   *Parikh, P. J., Meller, R.D.*: A note on worker blocking in narrow-aisle order picking systems when pick time is non-deterministic. IIE Transactions 42 (2010), pp. 392-404.

[9]   *Ratliff, H. D., Rosenthal, A. S.:* Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. Operations Research 31 (1983), No. 3, pp. 507-521.

[10] *Roodbergen, K. J., de Koster, R.*: Routing methods for warehouses with multiple cross aisles. International Journal of Production Research 39 (2001), No. 9, pp. 1865-1883.

[11] *Roodbergen, K. J., de Koster, R.*: Routing order pickers in a warehouse with a middle aisle. European Journal of Operational Research 133 (2001), pp. 32-43.

[12] *Werth, F., Ullrich, O.:* Simulation ausgewählter Heuristiken zur Tourenplanung in manuellen Kommissionierstationen, In: ASIM-Treffen 2011 - Simulation technischer Systeme und Grundlagen und Methoden in Modellbildung und Simulation, Hrsg: Andreas Brenke. Krefeld: Shaker-Verlag, 2011, pp. 161-166.