A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete

Mustafa Misir[1,2], Wim Vancroonenburg[1,2], Katja Verbeeck[1,2], Greet Vanden Berghe[1,2]

[1] CODeS research group, KAHO Sint-Lieven
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
{mustafa.misir, wim.vancroonenburg, katja.verbeeck, greet.vandenberghe}@kahosl.be

[2] CODeS research group, Department of Computer Science, K.U.Leuven
Etienne Sabbelaan 53, 8500, Kortrijk, Belgium

Abstract

The existence of several solution strategies for different search and optimisation problems motivates the use of these strategies in a collaborative way. These collaborative approaches are considered in the context of hybridisation. Hyper-heuristics provide hybridisation by performing search on multiple low-level heuristics for finding efficient solutions in a problem-independent manner. In this study, a suite of simple moves is employed to solve real world instances of the ready-mixed concrete delivery problem. These basic moves are used under a selection hyper-heuristic that makes use of the new adaptive iteration limited list-based threshold accepting with a fixed limit and four others for comparison. These hyper-heuristics are tested on a set of real world ready-mixed concrete delivery problem instances and a performance analysis is carried out.

1 Introduction

The ready-mix concrete delivery problem is a complex vehicle routing and scheduling problem encountered in the construction sector. The problem considers scheduling deliveries of ready-mix concrete (RMC) from central production centres to customers' construction yards, using a large heterogeneous fleet of trucks. Due to the perishable nature of RMC, these deliveries must be performed under strict time constraints as the RMC may lose its quality (or even solidify) during transport. Furthermore, customers typically require a large amount of RMC that cannot be delivered by a single truck, necessitating multiple truck deliveries. To guarantee the quality of the concrete structure, it is important to avoid large time gaps between these deliveries as they are handled sequentially at a construction yard.

The concrete delivery problem represents main characteristics of both scheduling and routing problems. It has been looked at in a limited number of studies. A decision support system was developed to solve the ready-mixed concrete delivery problem consisting of routing for pumping vehicles and scheduling of concrete delivery vehicles [17]. In [11], genetic algorithms were used to determine the dispatching sequence of the trucks and a simulation technique was used for detecting time information regarding each operation. Different neural network models based on feed-forward and Elman recurrent networks were proposed in [12]. The problem was decomposed into the assignment of orders to the production centres and the assignment of trucks in [20, 21]. A two-phase approach involving genetic algorithms and a construction heuristic was designed. A mixed-integer programming model as well as a local search strategy capable of solving large real-world problem instances were studied in [1]. In [26], the problem was modelled as an integer multicommodity network flow problem while at the same time an improvement strategy using variable neighbourhood search was studied. In [25], a hybrid approach involving variable neighbourhood search and mixed integer linear programming supported by a very large neighbourhood search strategy was proposed.

Hyper-heuristics are easy-to-use high level search mechanisms aiming to solve any search and optimisation problem. They are expressed as problem-independent search mechanisms designed to manage a number of low-level search strategies [3]. The underlying motivation is to use the strength of each utilised search strategy. This approach is expected to provide performance improvement over the use of each method separately. Together with this high-level search approach, the problem-independent nature of hyper-heuristics is helpful to elevate their generality level. In the literature, several hyper-heuristics

have been developed and applied to various problems [4]. In [5], these hyper-heuristic methods were classified as *selection hyper-heuristics* and *generation hyper-heuristics*. Selection hyper-heuristics are composed from certain components to manage a set of low-level heuristics. These components involves some learning devices such as choice function [9, 22], reinforcement learning [19, 24], case-based reasoning [8] and learning classifier systems [16] as well some meta-heuristics like tabu search [13], simulated annealing [2, 7], late acceptance [23] and great deluge [14]. Generation hyper-heuristics focus on generating these low-level heuristics. Genetic programming [6] is the widely used option for such strategies.

In the present study, the ready-mixed concrete delivery problem consisting of a heterogeneous fleet of vehicles has been investigated. For solving this problem, a new hyper-heuristic involving an adaptive list-based acceptance mechanism is proposed. Besides the new hyper-heuristic, a group of hyper-heuristics composed of certain methods from the literature has been experimented with a number of real world problem instances and a performance analysis has been carried out. In the following section, the RMC problem will be explained in further detail. Then, in Section 3, a discussion about selection hyper-heuristics will be carried out. In the same section, the details of the proposed hyper-heuristic will also be presented. In Section 4, the experimental results will be analysed. The last section contains a conclusion and the discussion of possible future research.

2 The ready-mixed concrete delivery problem

The ready-mixed concrete (RMC) delivery problem considers a set of orders for RMC that need to be met by delivering concrete from a set of production centres, using a pool of vehicles. The dilemma here is to find a delivery schedule that completes all orders and minimises delays in deliveries, lags between deliveries, wasted RMC and transportation time. Furthermore, each delivery needs to respect some constraints related to the perishable nature of the RMC. Figure 1 shows an example of a simple RMC delivery problem. In what follows, we describe the three main elements of the real world problem that is subject of this paper.

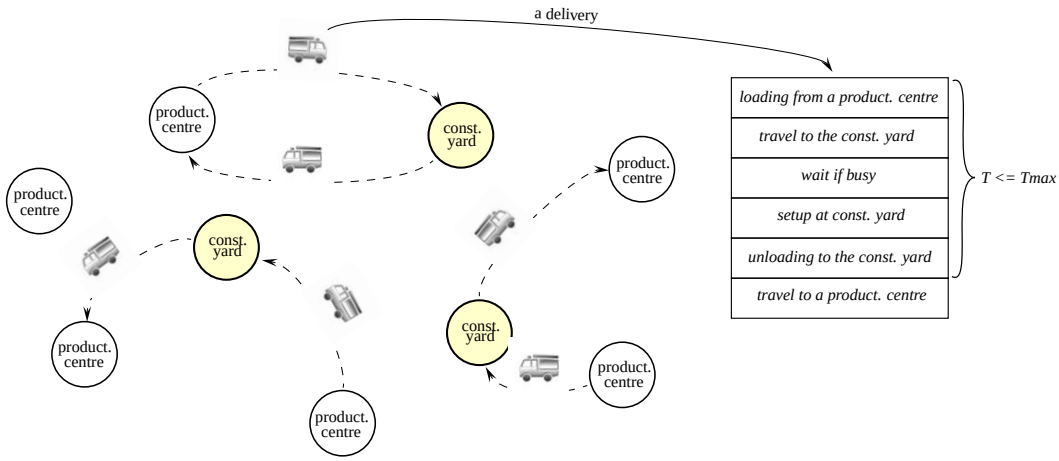

Figure 1: A RMC example

2.1 Production centres

Production centres are the locations where RMC is loaded into the trucks. Loading RMC requires a certain amount of time (constant), which is specific to the equipment used at the production centre.

2.2 Vehicles

Vehicles (trucks) are the main carriers of RMC that deliver the RMC at customers' construction yards. They are characterised by their volume, their pumpline length and their vehicle type. They can pick up concrete from any of the production centres. They are assumed to be present at the first production centre. Making a delivery follows the following sequence: 1) In this real-world problem, vehicles always take an amount of concrete that is equal to their capacity. Loading the concrete requires a certain amount of time that is specific to the production centre, as explained in Section 2.1. Furthermore, only one vehicle can perform loading operations at a production centre at any time, so loading operations should therefore not overlap. 2) Drive towards the construction yard. 3) Wait at the construction yard until the unloading can be set up. 4) Set up equipment at the construction yard. This takes an amount of time specific to the construction yard. This setup may overlap with a delivery already in progress. 5) Unload the RMC at the discharge rate specified by the customer. Only one vehicle can perform unloading at any time, so unloading operations may not overlap. 6) Return to a production centre for the next delivery.

Only a limited amount of time may pass between loading RMC at a production centre and unloading it at a construction yard, due to the perishable nature of the RMC. Therefore, each delivery must be finished with unloading after a specified amount of time, T_{max}. This possibly limits the amount of RMC that can be delivered by a vehicle, depending on the discharge rate used at the construction yard.

2.3 Orders

An order o is described by its order quantity q_o and a required discharge rate d_o at which the concrete is poured. This discharge rate determines the unloading time for a delivery: unloading a quantity q at a discharge rate d takes q/d time units. Each order has an earliest start time, before which no unloading operations can be performed. Some required properties for the delivery vehicles can be specified per order: 1) a customer can require a pumpline of a certain length for pouring concrete 2) certain vehicle types are prohibited by the customer. For example, some vehicle types might be too big to enter a customer's construction yard. Any vehicle matching the required properties for an order can deliver concrete to that customer.

2.4 Objective function

Symbol	Description
o	Order
d	Delivery
$\mathcal{O}$	Set of all orders
$\mathcal{D}$	Set of all deliveries
$\mathcal{D}_o$	Set of all deliveries for order o
C_o	Requested concrete amount for order o
$C_{d,o}$	Capacity of the truck handling delivery d for order o
W_o	Total waste for order o ($\sum_{d\in\mathcal{D}_o} C_{d,o} - C_o$)
$\mathcal{V}$	Set of all trucks
L_o	Lateness of the first delivery for order o
$t_{lag,d}$	Time lag for delivery d (time gap between unloading end time of previous delivery and unloading start time of delivery d)
$psOK_{o,d}$	is delivery d serviced from the preferred station for order o ($\in [0,1]$)
T_v	Total travelling time of a truck v
α_i	Weight for constraint i

Minimise:

$$\mathrm{f}_{objective} = \sum_{o\in\mathcal{O}} \left(\alpha_1 \cdot L_o + \alpha_2 W_o + \alpha_3 \sum_{d\in\mathcal{D}_o} psOK_{o,d} + \alpha_4 \sum_{d\in\mathcal{D}_o} t_{lag,d}\right) + \alpha_5 \sum_{v\in\mathcal{V}} T_v$$

The RMC problem considered in this paper aims at minimising the following objective function consisting of:

- the per-order lateness of the *first* delivery, weighted by α_1,

- the per-order wasted concrete, weighted by α_2; i.e. equal to the total delivered volume of RMC minus the ordered quantity,
- for each delivery, whether the RMC has been loaded at the preferred production centre, α_3,
- the per-order lag between subsequent deliveries, weighted by α_4,
- the total travelling time, weighted by α_5.

The weights of the objective function have been set after discussion with the company: $\alpha_1 = 10, \alpha_2 = 10, \alpha_3 = 1, \alpha_4 = 20, \alpha_5 = 20$. In addition, the maximum time for completing a delivery (T_{max}) is set to 100 minutes.

3 Selection hyper-heuristics

A traditional selection hyper-heuristic is composed of two sub-mechanisms, namely *heuristic selection* and *move acceptance*. A heuristic selection mechanism selects a low-level heuristic at each decision step. Using the selected heuristic, a new partial/complete solution is constructed/visited. Then, the quality of the resulting solution is examined relying on certain metrics by a move acceptance mechanism. After that, it is decided whether to accept or reject the new solution. Figure 2 demonstrates a single point search selection hyper-heuristic framework.

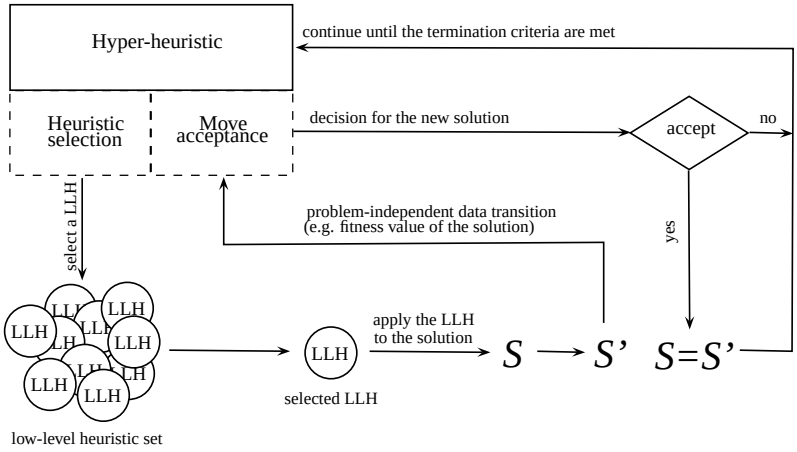

Figure 2: A traditional single-point search selection hyper-heuristic

For solving the RMC problem, a hyper-heuristic using the simple random (SR) selection mechanism [9] and a new move acceptance strategy, namely adaptive iteration limited list-based threshold accepting with a fixed limit (AILLA-F) was constructed. SR randomly chooses heuristics and AILLA-F accepts solutions which are visited by the selected heuristics using a list of previously found best solutions ($best_{list}$). The literature reports on similar strategies maintaining the history of fitness values as a list such as list-based threshold accepting [15] and late acceptance [23].

AILLA-F maintains a fixed-sized (l) list of previously visited new best solutions. Thus, whenever a new best solution is found, the oldest element in the list is removed and the new value is added the beginning of the list. The underlying idea is to use proper threshold values for different regions of the search space. An earlier version of AILLA-F [18] considered a threshold level determined by a constant value and by the current best solution. In AILLA-F, the list size is given as a parameter to reduce its user-dependency instead of using a constant value that directly affects the threshold value. Another critical component of AILLA-F is the iteration limit. It postpones the diversification decision by checking a fixed number of neighbouring solutions (k) before accepting a worsening one. When no new best solution is found, it is assumed that it is unlikely to be found during the search process in

Algorithm 1: AILLA-F move acceptance

Input: $i = 1, K \geq k \geq 0, l > 0$; $k = 5$,$K = 125$; $l = 10$
for *i=0* **to** *l-1* **do** $best_{list}(i) = f(S_{initial})$
if $adapt\_iterations \geq K$ **then**
  if $i < l - 1$ **then**
    $i++$
  end
end
if $f(S') < f(S)$ **then**
  $S \leftarrow S'$
  $w\_iterations = 0$
  if $f(S') < f(S_b)$ **then**
    $i = 1$
    $S_b \leftarrow S'$
    $w\_iterations = adapt\_iterations = 0$
    $best_{list}.remove(last)$
    $best_{list}.add(0, f(S_b))$
  end
else if $f(S') = f(S)$ **then**
  $S \leftarrow S'$
else
  $w\_iterations++$
  $adapt\_iterations++$
  if $w\_iterations \geq k$ *and* $f(S') \leq best_{list}(i)$ **then**
    $S \leftarrow S'$ and $w\_iterations = 0$
  end
end

progress. Then, a worsening solution is accepted based on the current threshold value. Moreover, if the employed threshold value is insufficient to discover new best solutions after K iterations, a larger value from the list is used as the threshold value. These threshold changes continue until the largest value in the list is being used or a new best solution is found. The pseudocode of AILLA-F is presented in Algorithm 1. S denotes the current solution, S' is the new solution generated after the selected heuristic is applied and S_b refers to the best solution found. The function f shows the quality of the given solution. $w\_iterations$ is the number of consecutively visited worsening solutions. $adapt\_iterations$ is a counter to decide whether the threshold level is increased.

Algorithm 2: Simulated annealing move acceptance

if $f(S') \leq f(S)$ **then**
  $S \leftarrow S'$
else if $rand(0,1) \leq \exp\Big[-\Big(f(S') - f(S)\Big)/(t_{remaining}/t_{total})\Big]$ **then**
  $S \leftarrow S'$
end

Algorithm 3: Great deluge move acceptance

if $f(S') \leq f(S)$ **then**
  $S \leftarrow S'$
else if $f(S') \leq f(S_{initial}) \times (t_{remaining}/t_{total})$ **then**
  $S \leftarrow S'$
end

Hyper-heuristics using the same selection mechanism with four other move acceptance mechanisms were used for comparison. These mechanisms are simulated annealing (SA), great deluge (GD), late acceptance (LATE) [10] and improving or equal (IE). IE accepts only improving and equal quality solutions. The rest of the hyper-heuristics also use IE. In addition, they may accept worsening solutions based on some time related threshold values, decreasing in time like SA and GD, or using gathered historical data during the run like LATE. The pseudocode of these move acceptance mechanisms are presented in Algorithm 2, Algorithm 3 and Algorithm 4 respectively. Regarding the SA and GD pseudocodes,

Algorithm 4: Late acceptance

Input: $list_{size} = 500$, $acc_{inx} = 0$
for *i=0* **to** $list_{size} - 1$ **do** *list(i) = f*($S_{initial}$)
if $f(S') \leq f(S)$ **then**
  $S \leftarrow S'$
  $list.exchange(acc_{inx}, f(S'))$
  $acc_{inx} + +$
else if $f(S') \leq list.get(acc_{inx})$ **then**
  $S \leftarrow S'$
  $list.exchange(acc_{inx}, f(S'))$
  $acc_{inx} + +$
else
  $list.exchange(acc_{inx}, (list_{min} + f(S))/2)$
  $acc_{inx} + +$
end
if $acc_{inx} = list_{size}$ **then**
  $acc_{inx} = 0$
end

$t_{remaining}$ refers to the remaining time and t_{total} shows the total execution time. In the pseudocode of LATE, $list.exchange(i, val)$ function exchanges the list value located at index i with a given value, val.

3.1 Initial solution

An initial solution is constructed at random. The process begins by sorting the orders based on their earliest start time in an increasing manner. While considering the orders according to this sorting, vehicles are randomly selected. If this vehicle is not assigned for a delivery yet, a starting station is selected. According to the required travelling time to the order's location and allowed transfer time limit, the maximum amount of concrete is determined. Using these random selections, new deliveries were introduced until all the requested concrete amount of the order is satisfied. The same procedure is repeated for the subsequent orders.

3.2 Low-level heuristics

For the RMC problem, 9 low-level heuristics were implemented. After applying a selected heuristic, a method to keep the resulting solution feasible is executed. This method simply reassigns time information whilst respecting T_{max} constraint and updates the number of deliveries if required.

- LLH_1: change return station of a vehicle that is responsible for a randomly selected delivery
- LLH_2: change vehicle of a randomly selected delivery
- LLH_3: ruin all the deliveries of an order and recreate them from scratch
- LLH_4: change the position of an order among the sequence of all orders
- LLH_5: move a delivery between deliveries belonging to an order
- LLH_6: change the amount to deliver for a randomly selected delivery
- LLH_7: change loading station for a randomly selected delivery
- LLH_8: swap two vehicles between two randomly selected deliveries belonging to a randomly selected order
- LLH_9: swap two vehicles between two randomly selected deliveries belonging to a two randomly selected orders

4 Experiments

The group of hyper-heuristics (SR-AILLA-F, SR-SA, SR-GD, SR-LATE, SR-IE) were applied 10-times to the constructed initial solutions belonging to 26 real-world RMC problem instances (Table 1) provided by ICORDA NV[1]. The total execution time was limited to 10 minutes for each run. The best two hyper-heuristics were additionally tested with 1 hour of execution time. The experiments were carried out using a Pentium Core 2 Duo 3 GHz PC with 3.23 GB memory.

Instances	# orders	Amount(m^3)	# vehicles	Instances	# orders	Amount(m^3)	# vehicles
p2011-01-10	6	80.75	28	p2011-02-01	18	299.85	28
p2011-01-11	17	598.75	34	p2011-02-02	16	333.25	28
p2011-01-12	26	748.25	34	p2011-02-03	19	422	28
p2011-01-13	15	581.95	31	p2011-02-07	17	317.25	28
p2011-01-14	21	474.95	32	p2011-02-08	14	233.5	31
p2011-01-18	17	395.25	28	p2011-02-09	17	588.25	28
p2011-01-19	18	436	28	p2011-02-10	16	507	28
p2011-01-20	14	309.25	28	p2011-02-11	19	651.25	30
p2011-01-21	17	308	28	p2011-02-14	15	233.95	32
p2011-01-25	13	280.25	28	p2011-02-15	24	1195.7	28
p2011-01-26	15	389.8	28	p2011-02-16	20	554.6	28
p2011-01-27	19	610.25	28	p2011-02-18	24	796.5	28
p2011-01-28	19	546.5	29	p2011-02-21	19	747.25	30

Table 1: The details of the RMC instances

4.1 Computational results

Table 2 shows the average fitness values of each hyper-heuristic on the tested instances. The hyper-heuristics are ranked as SR-AILLA-F, SR-LATE, SR-SA, SR-IE and SR-GD from the best to the worst based on their average performances. The table also clearly indicates that the two best hyper-heuristics use list-based threshold accepting strategies. In its turn this shows that the study of the hyper-heuristics' past behaviour can be helpful in pronouncing further judgements on diversification.

Instances	SR-AILLA-F		SR-LATE		SR-SA		SR-GD		SR-IE	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
p2011-01-10	**3783.85**	0.52	3829.15	145.50	3874.75	192.90	3828.75	2105.76	3837.65	696.88
p2011-01-11	**42308.55**	517.84	42547.42	818.30	43376.02	1092.83	44097.92	53050.34	43054.42	49643.52
p2011-01-12	298283.35	15846.13	366837.02	37818.51	**270496.22**	23940.32	359933.61	23268.58	314874.22	29613.89
p2011-01-13	**33431.68**	1549.30	33787.85	1583.88	34073.59	1610.74	36643.53	3912.76	34654.69	2065.12
p2011-01-14	30653.32	1689.16	**30185.73**	1538.58	30804.45	2017.61	41820.34	3641.37	33580.48	3360.66
p2011-01-18	**68635.28**	3094.89	69583.50	2193.09	73818.34	3906.36	78536.53	138634.46	74115.77	4140.76
p2011-01-19	**95806.79**	14695.65	110216.07	6790.16	102069.11	9294.39	132954.89	8088.98	102966.30	8758.61
p2011-01-20	94802.50	4333.84	**90405.56**	5608.52	100130.03	8602.82	110974.18	4917.97	109269.06	11244.14
p2011-01-21	62751.37	1174.61	**62518.80**	1751.25	64076.47	2338.30	64976.16	1120.64	64783.31	3250.76
p2011-01-25	50591.05	2643.41	**48687.56**	1758.78	52140.23	1826.53	55967.50	112545.13	55205.67	2969.21
p2011-01-26	**71658.31**	2093.15	72209.63	1559.61	76691.32	2836.23	81691.65	94452.10	79853.11	112677.12
p2011-01-27	**78696.14**	4434.09	80972.00	1846.89	87373.60	9548.44	91418.55	3888.31	83162.08	2819.35
p2011-01-28	**40009.56**	1879.02	40461.41	1461.14	44268.95	2200.93	55732.31	4748.82	46977.88	3606.37
p2011-02-01	**38790.96**	1718.79	38967.25	1736.38	41940.66	2908.43	49864.53	2050.40	45459.96	3542.27
p2011-02-02	**24692.87**	550.05	24992.28	518.33	25171.48	508.80	24943.45	15600.81	25693.68	18044.14
p2011-02-03	**65749.17**	1745.81	65942.09	1493.24	67370.82	2444.06	73484.04	2072.34	70201.96	5123.40
p2011-02-07	**34654.58**	1010.02	34700.01	925.78	35017.62	1516.60	37427.25	89373.26	36707.73	78029.03
p2011-02-08	**18553.10**	87.29	18569.00	197.88	18676.10	316.24	18677.68	148.08	18855.70	288.38
p2011-02-09	39427.22	1179.49	**39068.37**	1194.40	40534.73	1763.98	39543.35	1266.16	39884.30	1156.13
p2011-02-10	66574.60	70.22	67437.70	779.76	**66513.10**	41.43	69441.40	23206.06	66550.80	24954.80
p2011-02-11	**51853.51**	3449.69	53825.82	2972.18	55322.35	4904.08	67621.69	3337.41	58926.42	4519.54
p2011-02-14	**59601.56**	2950.68	60426.73	1718.60	66759.76	9968.05	67664.46	2824.46	69954.17	8374.82
p2011-02-15	**591919.15**	26627.67	660075.33	22085.79	592718.33	41801.29	627683.22	30100.45	596502.97	48846.38
p2011-02-16	**104711.29**	3967.35	110962.91	4177.45	111031.32	7179.41	123034.98	12367.71	119981.34	9015.92
p2011-02-18	365809.87	61779.94	**357262.07**	11521.33	428190.88	38843.64	375636.81	419522.25	379902.35	578378.08
p2011-02-21	104091.79	15574.68	**92152.51**	3596.63	106756.11	12814.48	121561.69	11152.39	115964.24	10295.44

Table 2: Average fitness values with standard deviations for each hyper-heuristic

In Table 3, the top two hyper-heuristics from the 10-minutes experiments are tested with 1 hour of execution time. The test results indicate that for longer execution time, SR-AILLA-F still performs better than SR-LATE.

In Figure 3(a), changes on the best fitness values for the tested hyper-heuristics are illustrated. In these results, SR-AILLA-F, SR-SA and SR-IE improve the initial solution very fast in the beginning of

[1] www.icorda.be

Instances	SR-AILLA-F		SR-LATE	
	AVG	STD	AVG	STD
p2011-01-10	3783.75	0.85	**3782.55**	0.48
p2011-01-11	41457.12	299.84	**41325.45**	372.24
p2011-01-12	**217455.99**	20469.46	233072.96	18007.72
p2011-01-13	**31557.60**	1904.81	32171.19	2152.41
p2011-01-14	**27715.36**	2100.99	28106.35	2280.85
p2011-01-18	**64992.81**	1510.02	66909.32	719.75
p2011-01-19	**56455.19**	3325.83	59504.30	2387.96
p2011-01-20	**75576.76**	3438.29	83699.83	5580.40
p2011-01-21	**60749.08**	443.67	60806.60	902.09
p2011-01-25	**47507.55**	1864.27	48911.85	2855.42
p2011-01-26	**68304.51**	1781.66	70836.96	1576.90
p2011-01-27	**71668.21**	1851.01	73507.16	1801.41
p2011-01-28	**34126.37**	1144.11	35419.53	2063.34
p2011-02-01	**35766.23**	647.63	36733.65	1456.72
p2011-02-02	**24418.18**	140.39	24588.97	320.00
p2011-02-03	**61222.13**	1524.57	63064.78	2312.47
p2011-02-07	33841.77	455.49	**33766.98**	498.59
p2011-02-08	**18463.90**	47.64	18467.70	46.08
p2011-02-09	37987.95	472.21	**37635.25**	209.55
p2011-02-10	**66541.90**	57.23	67450.50	792.42
p2011-02-11	**49601.22**	553.27	50239.25	1472.44
p2011-02-14	**54578.45**	1187.21	56709.26	1558.92
p2011-02-15	500672.21	14764.59	**480298.53**	7223.04
p2011-02-16	**91456.18**	1608.41	92431.51	2230.97
p2011-02-18	300698.23	33498.32	**279121.50**	4453.98
p2011-02-21	68985.05	5659.49	**72281.96**	6161.38

Table 3: Average fitness values with standard deviations for the top two hyper-heuristics with 1 hour of execution time

the search. SR-LATE is relatively slower. For SR-GD, the improvement process lags behind compared to the other methods, but it consistently improves the solution.

During the further iterations, the performance of the hyper-heuristics changes. SR-LATE and SR-GD catch the other hyper-heuristics. SR-IE almost stops improving after first 200 seconds due to the lack of an effective diversification component.

Figure 3(b) presents the heuristics that found new best solutions during the run for a particular instance. From the figure, it can be deduced that each heuristic contributes to the improvement process. However, their effect might change for different parts of the search space. Therefore, employing an intelligent selection mechanism can provide further improvements.

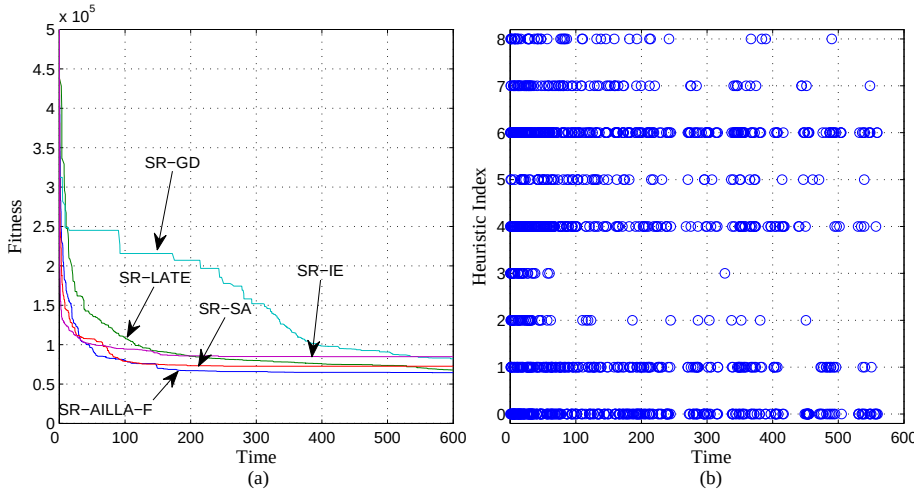

Figure 3: (a) The best fitness changes during the run on p2011-01-18 (run 7), (b) The heuristics that found new best solutions during the run on p2011-02-15 (run 5)

5 Conclusion

Hyper-heuristics are effective management strategies aiming to solve various problems in a fast and easy way. Therefore, different hyper-heuristics have been developed and applied to solve distinct problems.

In this study, we employed a hyper-heuristic using simple random as a selection mechanism and a new move acceptance strategy, i.e. adaptive iteration limited list-based threshold accepting with a fixed limit. For the purpose of comparison, the proposed hyper-heuristic together with four other hyper-heuristics from the literature were applied to a set of ready-mixed concrete delivery problem instances. These four approaches consist of the same selection mechanism with late acceptance, simulated annealing, great deluge, improving equal move acceptance mechanisms. The experimental results showed that the proposed approach performs better than the other hyper-heuristics based on their performance over the target instances. Furthermore, extended execution time limit did not change the performance gap.

In the future, intelligent collaborations between hyper-heuristic components will be investigated. For the selection process, a learning based approach will be designed and combined with the proposed acceptance mechanism. In addition, the user-dependency of the required parameters for the acceptance strategy will be reduced. The generality and robustness of the hyper-heuristic will be examined via extensive experiments additionally on other combinatorial optimisation problems.

References

[1] L. Asbach, U. Dorndorf, and E. Pesch. Analysis, modeling and solution of the concrete delivery problem. *European Journal of Operational Research*, 193(3):820–835, 2009.

[2] R. Bai and G. Kendall. An investigation of automated planograms using a simulated annealing based hyper-heuristics. In *Meta-heuristics: Progress as Real Problem Solvers, Selected Papers from the 5th MIC*, pages 87–108, 2005.

[3] E.K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg. *Handbook of Meta-Heuristics*, chapter Hyper-Heuristics: An Emerging Direction in Modern Search Technology, pages 457–474. Kluwer Academic Publishers, 2003.

[4] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. A survey of hyper-heuristics. Cs technical report no: Nottcs-tr-sub-0906241418-2747, Univeristy of Nottingham, 2009.

[5] E.K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J.R. Woodward. A classification of hyper-heuristic approaches. *Handbook of Metaheuristics*, pages 449–468, 2010.

[6] E.K. Burke, M.R. Hyde, and G. Kendall. Evolving bin packing heuristics with genetic programming. In *Proceedings of the 9th Parallel Problem Solving from Nature (PPSN'IX)*, volume 4193 of *LNCS*, pages 860–869, 2006.

[7] E.K. Burke, G. Kendall, M. Misir, and E. Ozcan. Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, pages 1–18, 2010. 10.1007/s10479-010-0782-2.

[8] E.K. Burke, B. MacCarthy, S. Petrovic, and R. Qu. Knowledge discovery in a hyperheuristic using case-based reasoning on course timetabling. In *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT'02)*, pages 276–286, Gent, Belgium, August 21–23 2002.

[9] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *PATAT '00: Selected papers from the Third International Conference on Practice and Theory of Automated Timetabling III*, pages 176–190, London, UK, 2001. Springer-Verlag.

[10] P. Demeester, P. De Causmaecker, and G. Vanden Berghe. A general approach for exam timetabling: a real-world and a benchmark case. In *Proceedings of the 8th International Conference on the Practice and Theory of Automated Timetabling (PATAT'10)*, Belfast, Northern Ireland, August 10–13 2010.

[11] C.W. Feng, T.M. Cheng, and H.T. Wu. Optimizing the schedule of dispatching RMC trucks through genetic algorithms. *Automation in Construction*, 13(3):327–340, 2004.

[12] L.D. Graham, D.R. Forbes, and S.D. Smith. Modeling the ready mixed concrete delivery system with neural networks. *Automation in Construction*, 15(5):656–663, 2006.

[13] G. Kendall and N.M. Hussin. An investigation of a tabu-search-based hyper-heuristic for examination timetabling. In *Multidisciplinary scheduling: theory and applications: 1st International Conference, MISTA'03: Nottingham, UK, 13-15 August 2003: selected papers*, page 309. Springer Verlag, 2005.

[14] G. Kendall and M. Mohamad. Channel assignment optimisation using a hyper-heuristic. In *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS'04)*, pages 790–795, Singapore, December 1–3 2004.

[15] D.S. Lee, V.S. Vassiliadis, and J.M. Park. List-based threshold-accepting algorithm for zero-wait scheduling of multiproduct batch plants. *Industrial & Engineering Chemistry Research*, 41(25):6579–6588, 2002.

[16] J.G. Marn-Blzquez and S. Schulenburg. A hyper-heuristic framework with xcs: Learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients. In *IWLCS*, volume 4399 of *LNCS*, pages 193–218, 2007.

[17] N.F. Matsatsinis. Towards a decision support system for the ready concrete distribution system: A case of a Greek company. *European Journal of Operational Research*, 152(2):487–499, 2004.

[18] M. Misir, K. Verbeeck, P. De Causmaecker, and G. Vanden Berghe. Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'10)*, pages 2875–2882, Barcelona, Spain, July 18–23 2010.

[19] M. Misir, T. Wauters, K. Verbeeck, and G. Vanden Berghe. A new learning hyper-heuristic for the traveling tournament problem. In *Proceedings of the 8th Metaheuristic International Conference (MIC'09)*, Hamburg, Germany, July 13–16 2009.

[20] D. Naso, M. Surico, B. Turchiano, and U. Kaymak. Just-in-time production and delivery in supply chains: a hybrid evolutionary approach. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 2, pages 1932–1937. IEEE, 2005.

[21] D. Naso, M. Surico, B. Turchiano, and U. Kaymak. Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete. *European Journal of Operational Research*, 177(3):2069–2099, 2007.

[22] E. Ozcan, B. Bilgin, and E.E. Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23, 2008.

[23] E. Ozcan, Y. Bykov, M. Birben, and E.K. Burke. Examination timetabling using late acceptance hyper-heuristics. In *Proceedings of Congress on Evolutionary Computation (CEC'09)*, 2009.

[24] E. Ozcan, M. Misir, G. Ochoa, and E.K. Burke. A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing*, 1(1):39–59, 2010.

[25] V. Schmid, K.F. Doerner, R.F. Hartl, and J.J. Salazar-González. Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers & Operations Research*, 37(3):559–574, 2010.

[26] V. Schmid, K.F. Doerner, R.F. Hartl, M.W.P. Savelsbergh, and W. Stoecher. A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1):70, 2009.