Analysing a Hybrid Model-Based Evolutionary Algorithm for a Hard Grouping Problem

Sebastian Raggl[1(✉)], Andreas Beham[1,2], Stefan Wagner[1], and Michael Affenzeller[1,2]

[1] Heuristic and Evolutionary Algorithms Laboratory, University of Applied Sciences Upper Austria, Hagenberg, Softwarepark 11, 4232 Hagenberg, Austria
sebastian.raggl@fh-hagenberg.at

[2] Institute for Formal Models and Verification, Johannes Kepler University Linz, Altenberger Straße 69, 4040 Linz, Austria

Abstract. We present a new hybrid model-based algorithm called Memetic Path Relinking (MemPR). MemPR incorporates ideas of memetic, evolutionary, model-based algorithms and path relinking. It uses different operators that compete to fill a small population of high quality solutions. We present a new hard grouping problem derived from a real world transport lot building problem. In order to better understand the algorithm as well as the problem we analyse the impact of the different operators on solution quality and which operators perform best at which stage of optimisation. Finally we compare MemPR to other state-of-the-art algorithms and find that MemPR outperforms them on real-world problem instances.

Keywords: Hybrid algorithm · Memetic algorithm
Estimation of distribution algorithm · Grouping problem

1 Introduction

Many modern optimisation algorithms are actually hybrids between different techniques. Memetic algorithms are the combination of population-based and trajectory based optimisation techniques. They have attracted a significant amount of research interest in recent years [1]. Memetic algorithms are especially successful at solving combinatorial optimisation problems [2]. Estimation of distribution algorithms are another class of algorithms that have been studied extensively [3]. They work by iteratively learning a statistical model of good solutions followed by the sampling of new solutions from that model. Path relinking (PR) was originally employed as an intensification strategy for tabu search [4]. It was also successfully used to explore paths between elite solutions in GRASP and scatter search [5,6].

The no free lunch theorem states that no algorithm can outperform all others on every problem [7]. The main idea behind the algorithm presented in this paper

R. Moreno-Díaz et al. (Eds.): EUROCAST 2017, Part I, LNCS 10671, pp. 347–354, 2018.
https://doi.org/10.1007/978-3-319-74718-7_42

is that by incorporating all of the previously mentioned approaches we can get a very robust algorithm that can tackle a wide variety of problem instances with different characteristics. An algorithm that treats problems as a black box and can solve them reasonable well is very valuable, especially when dealing with real-world problems.

1.1 Real-World Problems

When optimising real-world problems it is very likely that the problem definition, i.e. constraints, objective, is going to be adapted over time. When presenting optimisation results of real world problems to practitioners it is very important that the presented solutions are at least local optima. Otherwise even very good results might be dismissed when a human can find, inspired by the presented solution, a very similar but better solution.

The rest of this article is structured as follows. In Sect. 2 we describe the real-world transport lot building problem we want to solve. In Sect. 3 we describe our new memetic path relinking algorithm. In Sect. 4 we analyse how much the different operators contribute to the population and how this contributions change over time. Additionally we investigate the influence of the operators on the quality of solutions found. We compare our algorithm against four different algorithms on real-world problem instances in Sect. 5. Finally, there is a short discussion and outlook.

2 Problem

The problem is concerned with grouping items into transport lots while respecting restrictions on which items can be transported together as well as dependencies between items. There is a set of n items $\mathcal{I} = \{i_1, ..., i_n\}$ that need to be grouped together into an ordered set of groups $\mathcal{S}$. The groups have a maximum size of N. The relationships between the items are described by an undirected weighted graph $G_p = (\mathcal{I}, \mathcal{R})$. The vertices are items and the weight of the edges are the costs of putting two items into the same group. There is a directed graph $G_d = (\mathcal{I}, \mathcal{D})$ with the items as vertices and edges describing dependencies between items. If an item a depends on another item b, item a must be either in the same group as b, or in a group with a higher index in $\mathcal{S}$. Such a dependency is modelled as an edge in the dependency graph $(a, b) \in D$.

$$\min |S| + \sum_{s \in \mathcal{S}} C(s) \tag{1}$$

$$s.t.\ (a, b) \in R \qquad \forall_{s \in \mathcal{S}}\ \forall_{a \in s,\ b \in s} \tag{2}$$

$$S(a) \leq S(b) \qquad \forall_{(a,b) \in \mathcal{D}} \tag{3}$$

$$|s| \leq N \qquad \forall_{s \in \mathcal{S}} \tag{4}$$

We want to find the assignment to the smallest number of groups with the lowest total cost. The function $C(\mathcal{S}) \rightarrow \mathbb{R}$ calculates the costs of a group using

the weights on G_p. If an edge in G_p has a high weight placing the two items connected by that edge in the same group is unfavourable. Two items that do not share an edge in G_p must not be in the same group which is ensured by constraint 2. Constraint 3 guarantees that every item that depends on another one is either in the same group or in a group with a bigger index than the item it depends on. The function $S(\mathcal{I}) \rightarrow \mathbb{Z}$ maps an item to the index of the group it belongs to. The order of groups that do not depend on each other is irrelevant. A good way to check constraint 3 is to translate the item dependency graph into a group dependency graph for a given grouping and then check that this graph is acyclic. If this is the case, any topological ordering of the groups in the group dependency graph fulfils the constraint.

In order to be able to use single objective optimisation algorithms we model the constraints as soft constraints by adding a penalty for every constraint that is violated.

3 Memetic Path Relinking

Memetic Path Relinking (MemPR) is a new hybrid model-based algorithm based on the observations made above. The MemPR framework can be applied to a wide variety of problems thanks to the three variants using different solution encodings. The first variant optimises binary vectors, the second operates on permutations and the third can solve grouping problems by using the linear linkage encoding [12]. In this paper we are interested in the latter variant.

MemPR is designed to be used for black box optimisation and in order to be able to tackle problems with wildly different characteristics it does not rely on a single set of operators but instead uses six competing approaches for generating new individuals. MemPR uses the following operators:

- *Creation heuristic*
  MemPR can incorporate arbitrary creation heuristics but in the black box optimisation use case random solutions are sampled from the entire solution space.
- *Breeding*
  The breeding operator randomly selects two parents from the population and produces N children using any of the available crossover operators, where N is the number of dimensions the problem has. The best individual among these children is further improved by local search and reported as the result of the operator.
- *Path-relinking*
  This operator searches a path in the solution space between two randomly selected solutions and reports the best solution found on the path [4].
- *Path-delinking*
  The delinking procedure works the same as relinking but instead of a path starting from one solution and getting increasingly similar to a target solution it searches a path that leads away from the target solution.

- *Sampling from a bivariate model*
  The sampling operator builds a model of the probability of two items belonging to the same group using the current population. This model can subsequently be used to sample new solutions. Similarly to the breeding operator multiple samples are produced and the best one is returned.
- *Hill climbing*
  Different types of local search can be used, but we found that first improvement local search is well suited because it uses fewer evaluations.
- *Adaptive walk*
  A local search that may also accept steps that reduce quality is used to escape local optima.

Algorithm 1 shows how MemPR combines the operators described above. The population initially only consists of two individuals and can grow up to a predetermined size, usually twenty individuals. Those first two individuals are constructed by the creation heuristics and then improved using local search. In every iteration the breed, sample, relink and delink operators create new individuals and try to insert them into the population. Only if they are all unsuccessful is an adaptive walk used, to try to escape one of the local optima the population is comprised of.

Algorithm 1. Memetic Path Relinking

```
pop ← [HillClimb(Create()), HillClimb(Create())]
while not termination criteria reached do
  improved ← Replace(pop, Breed())
  improved ← Replace(pop, Relink()) or improved
  improved ← Replace(pop, Delink()) or improved
  sample ← Sample(pop)
  improved ← Replace(pop, sample) or improved
  if not improved and not Replace(pop, HillClimb(sample)) then
    Replace(pop, AdaptiveWalk())
  end if
end while
return best(pop)
```

Due to the small population size it is very important to maintain enough diversity so that the population does not prematurely converge. The replacement procedure is a central piece of MemPR in that regard. It aims to keep the population as diverse as possible by using the following rules. As long as the population has not reached its maximum size new individuals are simply added to the population. If the maximum population size is reached, and the new solution is on a plateau, meaning there are two ore more solutions with the same fitness, the solution to be replaced is chosen such that the average distance within the plateau is maximised. If the new individual is not on a plateau, it replaces the member of the population that has lower or equal fitness and is most similar to the new one. If no suitable candidate could be found replacement fails.

4 Analysing the Operators

Given that we have multiple operators that all produce individuals competing to be part of a small population. It is interesting to compare the success rates of the operators. In all our tests we use real-world problem instances with 53, 54, 64, 85, 101, and 145 items.

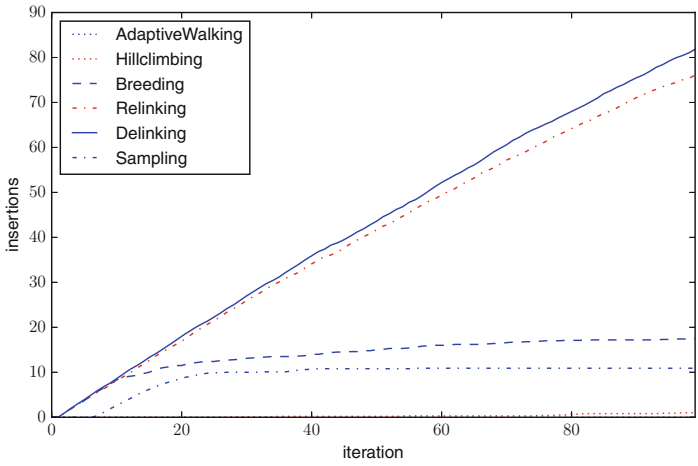

Fig. 1. Average number of individuals successfully inserted per operator into the population in the first 100 iterations

Figure 1 shows how often the different operators on average manage to create a solution that is accepted into the population. The delinking operator has the highest acceptance rate. This is explained by the fact that the replacement procedure favours a diverse population. Relinking has a slightly lower success rate. Breeding and sampling initially are as successful but fall behind as the optimisation run progresses. Both these operators essentially try to combine the groups found in individuals in the population into new individuals. Because of the nature of the problem a solution produced this way has a high chance of violating one of the constraints described in Sect. 2. The hill climber that is applied to the best individual has a chance to fix the violations. Unfortunately there is no guarantee that another solution would not be a better starting point for the hill climber. The sampling operator is only applied once the population has reached its full capacity which is why there are no insertions in the first few iterations. Adaptive walking and hill climbing are only applied when none of the other operators were managed to add a new individual to the population and therefore are only active in later stages of an optimisation run.

In order to asses the impact the operators have on the solution quality we experiment with disabling different combinations of operators. The algorithm was executed ten times for every configuration tested with a time limit of ten

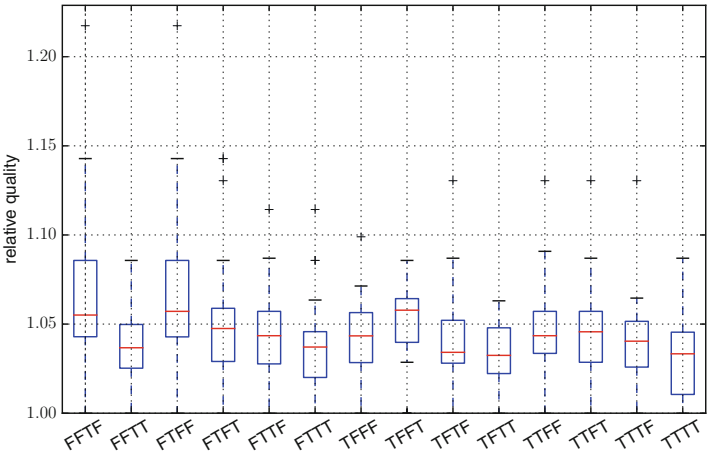

Fig. 2. Best relative quality achieved with different operator combinations

minutes per run. Figure 2 shows the best quality found for every combination relative to the best known quality. Four letter strings are used to signify which operator is enabled. The first letter is T, for true, if breeding is enabled and F, for false, if it is not. The second letter stands for delinking, the third for relinking and the last one for sampling.

The variant that uses only the sampling operator (FFFT) is excluded from the graph because it performs much worse. Between the variants only using a single operator the one using breeding performs best. Which is interesting considering that we showed earlier that the re/delinking are more likely to produce solutions that are integrated into the population. There are clearly interactions between the different operators, for example both relinking and sampling alone are not particularly good, but when combined into the FFTT variant they are quite competitive. On the other hand adding sampling to the breeding only variant decreases solution quality. The variants using all operators (TTTT) and the one only disabling delinking (TFTT) have similar and very good average solution qualities. The variant using delinking does however have the best median quality, because delinking helps preserve population diversity.

5 Algorithm Comparison

We compare MemPR with four algorithms available in the HeuristicLab optimisation environment [8] on real-world problem instances of various sizes. The algorithms we compare against are Variable Neighbourhood Search (VNS) [11], Evolution Strategy (ES), Offspring Selection Genetic Algorithm (OSGA) [10] and Age Layered Population Structure Genetic algorithm (ALPS) [9]. All algorithms use linear linkage encoding and randomly choose between greedy partition crossover, group crossover, lowest index max crossover and lowest index

first crossover [12,13]. MemPR and VNS both need a definition of neighbourhood and the both use the same definition in our tests. The neighbourhood of a solution are all solutions reached by applying any move out of four different types of moves. The first two types are extracting an item from a group and either inserted into an existing or a new group. The remaining moves either split a group into two smaller groups, or merge two groups into one.

Table 1. Best quality found by different algorithms

Items	MemPR		ALPS		ES		OSGA		VNS	
	avg	std	avg	std	avg	std	avg	std	avg	std
53	15.00	0.00	15.00	0.00	16.10	0.00	16.80	2.25	19.80	92.25
54	**15.90**	0.32	16.00	0.00	16.00	0.32	16.00	0.63	20.00	2.39
64	**16.48**	0.33	16.51	0.10	17.06	0.11	16.67	0.39	17.19	0.29
85	**21.70**	0.29	22.40	0.82	24.30	0.13	22.46	0.34	27.67	2.35
101	**23.10**	0.32	24.30	0.48	34.10	31.94	28.20	2.35	95.10	53.33
145	**35.30**	0.48	36.20	0.42	39.50	0.71	52.70	32.10	100.10	34.49

Table 1 shows the average best quality reached in ten repetitions with an evaluation budget of 10^7. The parameters of the algorithms were tuned by hand. On the smallest instances ALPS and ES achieves comparable solutions, but MemPR consistently outperforms the other algorithms on the bigger instances. While the differences between MemPR and ALPS are numerically not very big, in the real world application saving a single transport is significant. The three population based evolutionary algorithms manage find better solutions than VNS. ALPS and OSGA perform roughly the same on the first four instances, on the bigger instances OSGA struggles with premature convergence while ALPS, which is designed to avoid this problem, does not.

6 Conclusion

We present a new memetic algorithm that incorporates crossover operators and local search with path-relinking and sampling from a model. From both the comparison with other algorithms as well as the analysis of the operators we conclude that combinations of the different operators performs better than every individual operator. We further observed that preserving diversity during the search is vital for finding good solutions for our real world grouping problem. The delinking operator together with the replacement strategy manage to preserve diversity. This leads to good performance compared to other algorithms tested.

Acknowledgements. The work described in this paper was done within the COMET Project Heuristic Optimization in Production and Logistics (HOPL), #843532 funded by the Austrian Research Promotion Agency (FFG).

References

1. Amaya, J.E., Cotta Porras, C., Fernández Leiva, A.J.: Memetic and hybrid evolutionary algorithms. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 1047–1060. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-43505-2_52
2. Merz, P.: Memetic algorithms and fitness landscapes in combinatorial optimization. In: Neri, F., Cotta, C., Moscato, P. (eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence, vol. 379, pp. 95–119. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-23247-3_7
3. Pelikan, M., Hauschild, M.W., Lobo, F.G.: Estimation of distribution algorithms. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 899–928. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-43505-2_45
4. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Norwell (1997)
5. Glover, F., Laguna, M., Martí, R.: Fundamentals of scatter search and path relinking. Control cybern. **29**, 653–684 (2000)
6. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 57, pp. 219–249. Springer, Boston (2003). https://doi.org/10.1007/0-306-48056-5_8
7. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
8. Wagner, S., Kronberger, G., Beham, A., Kommenda, M., Scheibenpflug, A., Pitzer, E., Vonolfen, S., Kofler, M., Winkler, S., Dorfer, V., Affenzeller, M.: Architecture and design of the heuristiclab optimization environment. In: Klempous, R., Nikodem, J., Jacak, W., Chaczko, Z. (eds.) Advanced Methods and Applications in Computational Intelligence. Topics in Intelligent Engineering and Informatics, vol. 6, pp. 197–261. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-319-01436-4_10
9. Hornby, G.S.: ALPS: the age-layered population structure for reducing the problem of premature convergence. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, GECCO 2006, pp. 815–822. ACM, New York (2006)
10. Affenzeller, M., Wagner, S.: Offspring selection: a new self-adaptive selection scheme for genetic algorithms. In: Ribeiro, B., Albrecht, R.F., Dobnikar, A., Pearson, D.W., Steele, N.C. (eds.) Adaptive and Natural Computing Algorithms, pp. 218–221. Springer, Vienna (2005). https://doi.org/10.1007/3-211-27389-1_52
11. Mladenović, N., Hansen, P.: Variable neighborhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
12. Ülker, Ö., Özcan, E., Korkmaz, E.E.: Linear linkage encoding in grouping problems: applications on graph coloring and timetabling. In: Burke, E.K., Rudová, H. (eds.) PATAT 2006. LNCS, vol. 3867, pp. 347–363. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77345-0_22
13. Korkmaz, E.E.: Multi-objective genetic algorithms for grouping problems. Appl. Intell. **33**, 179–192 (2010)