Universidade Federal Fluminense
Instituto de Computação
Curso de Ciência da Computação

Daniel Paes Martins

Making Heuristics Faster with Data Mining

Niterói
2011

Daniel Paes Martins

Making Heuristics Faster with Data Mining

Monografia apresentada ao Curso de Ciência da Computação da UFF, como requisito para a obtenção parcial do grau de BACHAREL em Ciência da Computação.

Orientador: Alexandre Plastino de Carvalho

D.Sc.

Co-orientadora: Simone de Lima Martins

D.Sc.

Co-orientadora: Isabel Cristina Mello Rosseti

D.Sc.

Niterói

2011

Martins, Daniel

Making Heuristics Faster with Data Mining / Daniel Martins - 2011

xx.p

areas. I.Título.

CDU ???

Daniel Paes Martins

Making Heuristics Faster with Data Mining

Monografia apresentada ao Curso de Ciência da Computação da UFF, como requisito para a obtenção parcial do grau de BACHAREL em Ciência da Computação.

Aprovado em 09 de dezembro de 2011

Agradecimentos

Aos meus orientadores, pela oportunidade incrível de trabalhar com eles, e pelo apoio inestimável durante esse ano.

À minha mãe, por ter sempre me incentivado a me dedicar aos estudos.

Ao meu pai, por ter sempre me incentivado a buscar excelência em tudo que eu fizesse.

A toda a minha família, por ter sempre apoiado incondicionalmente as minhas decisões.

Ao querido amigo Hugo Barbalho, por todo apoio e auxílio.

Aos amigos Raphael Bottino, Cássio Fernando, Leonardo Freitas, e demais frequentadores do DACC, por terem me atrapalhado não só na confecção desse TCC, como em todas as minhas responsabilidades.

A todos os demais colegas da UFF, por terem contribuído para que esses anos de faculdade fossem tão agradáveis.

Resumo

Metaheurísticas representam uma importante classe de técnicas para a obtenção de boas soluções, em tempo computacional razoável, para problemas de otimização combinatória. São procedimentos de alto nível de propósito geral que podem ser instaciadas para explorar eficientemente o espaço de soluções de um problema combinatório específico. Um importante tópico em metaheurísticas é o desenvolvimento de metaheurísticas híbridas. Esses métodos híbridos resultam da combinação de conceitos e procedimentos de diferentes metaheurísticas clássicas ou da combinação de metaheurísticas com conceitos e processos de outras áreas de pesquisa responsáveis por realizar tarefas específicas que podem melhorar a técnica original. Anteriormente, foi desenvolvida uma versão da metaheurística GRASP (Greedy Randomized Adaptative Search Procedures) que incorporou um procedimento de mineração de dados. Considerou-se hipótese de que padrões obtidos por uma técnica de mineração de dados, a partir de um conjunto de soluções sub-ótimas de um problema de otimização combinatória, poderiam ser usados para guiar procedimentos metaheurísticos na busca por melhores soluções. Foram obtidos resultados promissores aplicando-se essas idéias a diferentes problemas de otimização combinatória, tais como: o problema do empacotamento de conjuntos, o problema de maximização da diversidade, e o problema de replicação de servidores para multicast confiável. O desafio deste trabalho é introduzir um procedimento de mineração de dados a um importante métdodo da literatura que combina elementos de diferentes metaheurísticas para resolver o problema das p-medianas, conhecido como heurística híbrida *multistart* – HH, buscando obter evidências de que, quando uma técnica é capaz de atingir a solução ótima, ou uma solução sub-ótima com pouca chance de melhora, os padrões minerados podem ser usados para guiar a busca pela solução ótima ou sub-ótima em menos tempo computacional. À nova versão denominou-se DM-HH. Experimentos computacionais, conduzidos em um conjuto de instâncias da literatura, mostraram que a nova versão da heurística híbrida foi capaz de atingir soluções ótimas e sub-ótimas, em média 27,32% mais rápido que a estratégia original.

Palavras chave: Metaheurísticas Híbridas, Problema das p-Medianas, Mineração de Dados.

Abstract

Hybrid metaheuristics – developed based on the combination of metaheuristics with concepts and techniques from other research areas – represent an important subject in combinatorial optimization research. Previously, a hybrid version of the GRASP (Greedy Randomized Adaptive Search Procedures) metaheuristic which incorporates a data mining procedure was developed and explored. The base hypothesis was that patterns obtained by a data mining technique, from a set of suboptimal solutions of a combinatorial optimization problem, could be used to guide metaheuristics procedures in the search for better solutions. Promising results were obtained when applying these ideas to different combinatorial problems, such as: the set packing problem, the maximum diversity problem and the server replication for reliable multicast problem. The challenge of this work is to introduce a data mining procedure into a state-of-the-art heuristic for a specific problem in order to give evidences that, when a technique is able to reach the optimal solution, or a near-optimal solution with little chance of improvements, the mined patterns could be used to guide the search for the optimal or near-optimal solutions in less computational time. We then developed a new version of a previously proposed and state-of-the-art multistart hybrid heuristic for the classical p-median problem, which combines elements of different traditional metaheuristics. Computational experiments, conducted on a set of instances from the literature, showed that the new version of the hybrid heuristic was able to reach optimal and near-optimal solutions, on average, 27.32% faster than the original strategy.

Keywords: Hybrid Metaheuristics, p-Median Problem, Data Mining.

Contents

List of Figures

List of Tables

1 Introduction

Metaheuristics represent an important class of techniques for obtaining good solutions, in reasonable time, for hard combinatorial optimization problems. They are general purpose high-level procedures that can be instantiated to explore efficiently the solution space of a specific optimization problem [23]. Tabu search, genetic algorithms, simulated annealing, ant systems and GRASP are examples of metaheuristics and have been applied to real-life problems of several areas of science over the last decades. An overview of heuristic search can be found in [30].

An important topic in metaheuristics research is the development of hybrid metaheuristics [36]. Such hybrid methods result from the combination of concepts and procedures of different classic metaheuristics or from the combination of metaheuristics with concepts and processes from other research areas responsible for performing specific tasks that can improve the original technique. An instance of the latter case, and subject of this work, is the hybrid version of a multistart heuristic that incorporates a data mining process.

Over the last years, a hybrid version of the GRASP metaheuristic, called DM-GRASP [33] was developed, which includes a data mining technique to improve the search through the solution space.

The GRASP (Greedy Randomized Adaptive Search Procedures) metaheuristic [5, 6] has been successfully applied to solve many combinatorial optimization problems, in several areas, such as scheduling, routing, partitioning, location and assignment [7, 8]. The solution search process employed by GRASP is performed iteratively and each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution and then its neighborhood is explored by the local search in order to find a better one. The result is the best solution found over all iterations.

Data mining refers to the automatic extraction of knowledge from datasets [16, 41]. The extracted knowledge, expressed in terms of patterns or rules, represents important features of the dataset at hand. Hence, data mining provides a means to better understand features implicit in raw data, which is fundamental in a decision-making process.

The hybridization of GRASP with a data mining process was first introduced and applied to the set packing problem [28, 29]. The basic idea was to obtain knowledge from the solutions obtained, in previous iterations, to guide the search for the next iterations. Patterns extracted from good quality solutions could be used to guide the search, leading to a more effective exploration of the solution space. The resulting method, the DM-GRASP metaheuristic, achieved successful results also when evaluated on two other combinatorial problems, namely, the maximum diversity problem [31] and the server replication for reliable multicast problem [32].

The idea of keeping track of recurrent good sub-solutions and fixing variables have been successfully explored coupled with other heuristics. Lin and Kernighan developed a heuristic for the travelling salesman problem [20], where they fix some links observed to occur in a number of previously locally optimum tours found by the algorithm. Lodi et al.[21] developed an evolutionary heuristic for quadratic 0-1 programming, where they present an intensification strategy used in a genetic algorithm to fix variables, which can have their values fixed during all steps of the algorithm or only during a given number of steps. Fleurent and Glover [9] described strategies for the quadratic assignment problem, in which, during the constructive procedure, they select elements to be inserted in a solution from an elite set containing the best solutions generated so far. Our idea is to use more elaborated techniques found in the data mining research area to search for good patterns extracted from a set of high quality solutions.

The previous DM-GRASP implementations were developed over a common framework, divided in two parts. In the first one, a number of GRASP iterations are executed and the best solutions are stored in an elite set. Then, a data mining algorithm is used to extract patterns from this set of sub-optimal solutions. In the second part, the GRASP iterations use the mined patterns to construct new solutions. In this framework, the data mining process is performed after exactly half of the GRASP iterations. According to the taxonomy of hybrid metaheuristics proposed in [36], the DM-GRASP framework can be classified as a high-level and relay hybrid metaheuristic. It is considered high-level, since the data mining technique and GRASP are self-contained and it is a relay hybridization because GRASP, the data mining process, and GRASP again are applied in a pipeline fashion.

The challenge of this work is to introduce a data mining procedure into a state-of-the-art heuristic for a specific problem in order to give evidences that, when a technique is able to reach the optimal solution, or a near-optimal solution with little chance of improvements, the mined patterns could be used to guide the search for the optimal or near optimal solutions in less computational time.

We chose, as the state-of-the-art algorithm to be the base of our study, the heuristic proposed in [26] for the classical p-median problem. It is a multistart hybrid heuristic which combines elements of different traditional metaheuristics. Computational experiments conducted on instances from the literature showed that this strategy was able to perform at least as well as other methods, and often better in terms of both running time and solution quality. The solutions obtained were always within 0.1% of the best known upper bounds.

Basically, this strategy is a multistart iterative method and each iteration constructs randomly a solution, which is then submitted to local search. In each iteration, the solution obtained by local search is combined with one solution from an elite set, made of the best reached solutions, through a path-relinking process [11]. After all iterations are concluded, a post-optimization phase is activated in which elite solutions are combined with each other. We will refer to this proposed strategy as Hybrid Heuristic (HH).

We then developed the Data Mining Hybrid Heuristic (DM-HH), introducing a data mining procedure into the HH implementation in order to explore patterns extracted from a set of good quality solutions. These patterns are used in the construction of new solutions, leading to a more effective search through the solution space. Computational experiments, comparing the HH and DM-HH strategies and conducted on a set of instances from the literature, showed that the new data mining version of the hybrid heuristic was able to reach optimal and near-optimal solutions, on average, 27.32% faster than the original strategy.

Another contribution of this work is to show that not only the traditional GRASP metaheuristic but also other more sophisticated heuristic, improved with a memory-based intensification mechanism, like the path-relinking technique, can benefit from the incorporation of a data mining procedure.

The remaining of this paper is organized as follows. In Section 2, we present the p-median problem and review the main concepts and the structure of the state-of-the-art Hybrid Heuristic for this combinatorial problem. The Data Mining Hybrid Heuristic, proposed in this work, is presented in Section 3. The computational experiments conducted to compare both strategies are reported and discussed in Section 4. In Section 5, we illustrate and justify the behavior of both strategies with some additional analysis. Finally, concluding remarks and some future works are pointed out in Section 6.

2 Multistart Hybrid Heuristic

As stated in [26], given a set F of m potential facilities, a set U of n customers, a distance function $d : U \times F \rightarrow \mathbb{R}$, and a constant $p \leq m$, the p-median problem consists of determining which p facilities to open so as to minimize the sum of the distances from each costumer to its closest open facility. It is a well-known NP-hard problem [19], with numerous applications in location science [37] and clustering [24, 39].

In [26], Resende and Werneck proposed a state-of-art multistart hybrid heuristic for the p-median problem, that combined elements of several traditional metaheuristics to find near-optimal solutions to this problem. Figure 2.1 summarizes this algorithm. Each iteration of this algorithm consists of the randomized construction of a solution (line 4), which is then submitted to local search (line 5). After this, a solution is chosen from the pool of elite solutions, made with some of the best solutions found in previous iterations (line 6), and is combined with the solution obtained by the local search through a process called path-relinking [11] (lines 7-10). Furthermore, after all iterations are completed, this algorithm executes the second phase, called post-optimization, in which elite solutions are combined with each other (line 13), and the best solution found after the post-optimization phase execution is taken as result.

The hybrid heuristic was compared with VNS (Variable Neighborhood Search) [17], VNDS (Variable Neighborhood Decomposition Search) [18], LOPT (Local Optimization) [35], DEC (Decomposition Procedure) [35], LSH (Lagrangean Surrogate Heuristic) [34], and CGLS (Column Generation with Lagrangean Surrogate Relaxation) [34]. Empirical results on instances from the literature attested the robustness of the hybrid algorithm, which performed at least as well as other methods, and often better in terms of both running time and solution quality. In all cases the solutions obtained by hybrid heuristic were within 0.1% of the best known upper bounds.

In the next subsections, we describe the main concepts and the structure of the hybrid heuristic.

```
procedure Hybrid_Heuristic($seed$,$maxit$,$elitesize$)
 1.  Randomize($seed$);
 2.  Init($elite\_set$,$elitesize$);
 3.  for $it \leftarrow 1$ to $maxit$ do
 4.    $S \leftarrow$ Construction_p_Median();
 5.    $S \leftarrow$ Local_Search_p_Median($S$);
 6.    $S' \leftarrow$ Select($elite\_set$);
 7.    if($S' \neq \varnothing$)
 8.      $S' \leftarrow$ Path_Relinking($S$,$S'$);
 9.      Update_Elite($elite\_set$,$S'$);
10.    end if
11.    Update_Elite($elite\_set$,$S$);
12.  end for;
13.  $S \leftarrow$Post_Optimization($elite\_set$);
14.  return $S$;
```

Figure 2.1: Pseudo-code of the hybrid heuristic.

2.1 Construction Phase

Figure 2.2 summarizes the construction method, which is based on the called standard greedy algorithm [4, 40]. This algorithm starts with the empty solution and adds facilities, one at time, choosing the most profitable in each iteration. The standard strategy is deterministic because it finds the same solution in all iterations. The construction algorithm of [26] is similar to this standard approach, but, instead of selecting the best among all possible options, it only considers $q < m$ possible insertions (line 5), chosen uniformly at random, in each iteration. The most profitable among these options is selected (line 6). The number $q = \lceil log_2(m/p) \rceil$ is chosen small enough to reduce the running time of the algorithm (when compared to the standard greedy) and to ensure a fair degree of randomization.

```
procedure Construction_p_Median()
1.  Init($n$,$m$,$p$);
2.  $q \leftarrow \lceil log_2(m/p) \rceil$;
3.  $S \leftarrow \varnothing$;
4.  for $it \leftarrow 1$ to $p$ do
5.    $RCL \leftarrow$ Create_RCL($q$);
6.    $u \leftarrow$ Best_Possibility($RCL$);
7.    $S \leftarrow S \cup \{u\}$;
8.  end for;
9.  return $S$;
```

Figure 2.2: Pseudo-code of the construction phase.

2.2 Local Search

The adopted local search procedure for the p-median problem, originally proposed by Teitz and Bart [38], is based on swapping facilities. Given an initial solution S, the procedure determines, for each facility $f \notin S$, which facility $g \in S$ (if any) would improve the solution the most if f and g were interchanged (i.e., if f were opened and g closed). If there is one such improving move, f and g are interchanged. The procedure continues until no improving interchange can be made, in which case a local minimum has been found.

2.3 Path-Relinking

Path-relinking was originally proposed by Glover [11] as an intensification strategy exploring trajectories connecting elite solutions obtained by tabu search or scatter search [10, 12]. The procedure starts by computing the symmetric difference between the two solutions, i.e., the set of moves needed to reach the solution target S_t from solution source S_s. The current solution S is initialized with S_s and a path of solutions is generated linking S_s and S_t. At each step, the procedure examines all moves from the current solution S and selects the one which results in the least cost solution. The best move is made and the set of available moves is updated. If necessary, the best solution along the path under construction is updated. The procedure terminates when S_t is reached and the best solution found after the path-relinking phase execution is taken as result.

2.4 Post-Optimization

The post-optimization phase in the hybrid heuristic combines the solutions of the pool of elite solutions to obtain even better ones. Each solution in the pool is combined with each other by path-relinking. The solutions generated by this process are added to a new pool of elite solutions, representing a new generation. The post-optimization algorithm proceeds, executing on the new generation, until it creates a generation that does not improve upon the previous one.

3 Data Mining Hybrid Heuristic

In this section, we propose a new version of the hybrid heuristic (HH), presented in the previous section, which incorporates a data mining process, called DM-HH, to solve the p-median problem. The basic concept of incorporating a data mining process is that patterns found in high quality solutions obtained in earlier iterations of the HH strategy can be used to conduct and improve the search process.

The DM-HH procedure is composed of two phases. The first one consists of executing $maxit/2$ pure HH iterations to obtain a set of different solutions. The d best solutions from this set of solutions compose the elite set for mining.

After this first phase, the data mining process is applied. It is responsible for extracting a set of patterns from the elite set. The patterns to be mined are sets of elements that frequently appear in solutions from the elite set. This extraction of patterns characterizes a frequent itemset mining application [16]. A frequent itemset mined with support $s\%$ represents a set of elements that occur in $s\%$ of the elite solutions.

Next, the second phase is performed. In this part, another $maxit/2$ slightly different HH iterations are executed. In these $maxit/2$ iterations, an adapted construction phase starts building a solution guided by a mined pattern selected from the set of mined patterns. Initially, all elements of the selected pattern are inserted into the partial solution, from which a complete solution will be built executing the standard construction procedure. This way, all constructed solutions will contain the elements of the selected pattern.

The pseudo-code of the DM-HH for the p-median problem is illustrated in Figure 3.1. In lines 2 and 3, the elite set of the original heuristic and the elite set for mining are initialized with the empty set. The loop from line 4 to line 15 corresponds to the first phase of the strategy, in which pure HH is performed for $maxit/2$ iterations. The original construction method is executed in line 5, followed by the local search method in line 6. In line 7, a solution is chosen from the pool of elite solutions of the original approach to be combined, in line 9, using the path-relinking process with the solution obtained by the local search. In lines 10 to 14, the elite set of the original algorithm and the elite set for mining, composed of d solutions, are updated with the solution obtained by the path-relinking process and with the solution obtained by the local search. In line 16, the data mining procedure extracts t patterns from the elite set, which are inserted in decreasing order of pattern size in the set of patterns. The loop from line 17 to line 27 corresponds to the second phase of the strategy. In line 18, one pattern is picked from the set of patterns in a round-robin way. Then the adapted construction procedure is performed in line 19, using the selected pattern as a starting point. In line 20, the local search is executed. After this, a solution is chosen from the pool of elite solutions of the original approach to be combined using the path-relinking with the solution obtained by the local search (lines 21 to 26). After all iterations are completed, this algorithm executes the post-optimization in line 28 and the best solution found after the post-optimization phase is taken as result.

In Figure 3.2, the pseudo-code of the adapted construction is illustrated. It is quite similar to the code described in Figure 2.2 with the difference that, instead of beginning the solution with an empty set, in line 3, it starts with all elements of the pattern supplied as a parameter. In the loop from line 4 to line 8, the solution is completed using the original construction method.

The extraction of patterns from the elite set, which is activated in line 16 of the pseudo-code presented in Figure 3.1, corresponds to the well-known frequent itemset mining (FIM) task. The FIM problem can be defined as follows.

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. A transaction is a subset of I and a dataset D is a set of transactions. A frequent itemset F, with support s, is a subset of I which occurs in at least $s\%$ of the transactions in D. The FIM problem consists of extracting all frequent itemset from a dataset D with a minimum support specified as a parameter. During the last two decades, many algorithms have been proposed to efficiently mine frequent itemsets [1, 13, 15, 22].

In this work, the useful patterns to be mined are sets of elements that commonly appear in sub-optimal solutions of the p-median problem. This is a typical frequent itemset mining application, where the set of items is the set of potential locations. Each transaction of the dataset represents a sub-optimal solution of the elite set. A frequent itemset mined from the elite set with support $s\%$ represents a set of locations that occur in $s\%$ of the elite solutions.

A frequent itemset is called maximal if it has no superset that is also frequent. In order to avoid mining frequent itemsets which are subset of one another, we decided to extract only maximal frequent itemset. To execute this task, we adopted the FPmax* algorithm [14], available at http://fimi.cs.helsinki.fi.

```
procedure DM_HH($seed$,$maxit$,$elitesize$,$t$)
1.  Randomize($seed$);
2.  $elite\_set \leftarrow \varnothing$;
3.  $elite\_set\_DM \leftarrow \varnothing$;
4.  for $it \leftarrow 1$ to $maxit/2$ do
5.    $S \leftarrow$ Construction_p_Median();
6.    $S \leftarrow$ Local_Search_p_Median($S$);
7.    $S' \leftarrow$ Select($elite\_set$);
8.    if($S' \neq \varnothing$)
9.      $S' \leftarrow$ Path_Relinking($S$,$S'$);
10.     Update_Elite($elite\_set$,$S'$);
11.     Update_Elite($elite\_set\_DM$,$S'$);
12.   end if
13.   Update_Elite($elite\_set$,$S$);
14.   Update_Elite($elite\_set\_DM$,$S$);
15. end for;
16. $patterns\_set \leftarrow$ Mine($elite\_set\_DM$, $t$);
17. for $it \leftarrow 1$ to $maxit/2$ do
18.   $pattern \leftarrow$ Select_Next_Largest_Pattern($patterns\_set$);
19.   $S \leftarrow$ Adapted_Construction_p_Median($pattern$);
20.   $S \leftarrow$ Local_Search_p_Median($S$);
21.   $S' \leftarrow$ Select($elite\_set$);
22.   if($S' \neq \varnothing$)
23.     $S' \leftarrow$ Path_Relinking($S$,$S'$);
24.     Update_Elite($elite\_set$,$S'$);
25.   end if
26.   Update_Elite($elite\_set$,$S$);
27. end for;
28. $S \leftarrow$ Post_Optimization($elite\_set$);
29. return $S$;
```

Figure 3.1: Pseudo-code of the DM-HH

```
procedure Adapted_Construction_p_Median($pattern$)
1. Init($n$,$m$,$p$);
2. $q \leftarrow \lceil log_2(m/p) \rceil$;
3. $S \leftarrow pattern$;
4. for $it \leftarrow |pattern| + 1$ to $p$ do
5.   $RCL \leftarrow$ Create_RCL($q$);
6.   $u \leftarrow$ Best_Possibility($RCL$);
7.   $S \leftarrow S \cup \{u\}$;
8. end for;
9. return $S$;
```

Figure 3.2: Pseudo-code of the adapted construction

4 Computational Experiments

In this section, the computational results obtained for HH and DM-HH are presented. The strategies were evaluated using three classes of instances. The first class, named ORLIB, consists of 40 instances and was taken from the OR-Library [3]. Each instance is a different graph with a corresponding value for p. The number of nodes (customers) varies from 100 to 900, and the value of p ranges from 5 to 200. The optimal values are known for these 40 instances. In the OR-Library, these 40 instances are identified by *pmed01* to *pmed40*.

Instances of the second class, named TSP, are sets of points on the plane. Originally proposed for the traveling salesman problem, they are available at the TSPLIB [25]. Every point is considered both a potential facility and a customer, and the cost of assigning customer c to facility f is simply the Euclidean distance between the points representing c and f (e.g. the costs are real values). From the TSP class, we considered the $FL1400$ instances, with 1400 nodes and with several different values for p (number of facilities to open).

The third class we study is named RW. Originally proposed in [27], it corresponds to completely random distance matrices. In every case, the number of potential facilities (m) is equal to the number of customers (n). The distance between each facility and each customer has an integer value taken uniformly at random from the interval $[1, n]$. Six different values of n were considered: 100, 250, 500, 1000, 1500, and 2000. In each case, several values of p were tested.

The algorithms were implemented in C++ and compiled with g++ (GCC) 4.2.3. The tests were performed on a 2.4 GHz Intel Core 2 Quad CPU Q6600 with 3 Gbytes of RAM, running Linux Kernel 2.6.24.

Both HH and DM-HH were run 9 times with a different random seed in each run. Each strategy executed 500 iterations. After having conducted some tuning experiments, we set some parameter values. The size of the elite set for mining (d) and the size of the set of patterns (t) were set to 10. And a set of facilities was considered a pattern if it was present in at least two of the elite solutions.

When executed for the 40 instances from the ORLIB class, both HH and DM-HH reached the optimal solution in all 9 runs. Table 4.1 presents the results related to execution time of both strategies. In this table, the first column presents the name, the number of customers and the value of p of the working instances, the second and fourth columns show the average execution time (in seconds) of HH and DM-HH, obtained for 9 runs, the third and fifth columns present the standard deviation value of these execution times. Smaller times are bold-faced. The sixth column shows the percentual difference between the HH and DM-HH average times in relation to the HH average time. In the last line, the average of the percentual differences is reported. We can observe that DM-HH was always faster than HH and that the standard deviations are quite small. On average, DM-HH was 25.06% faster than the HH strategy for the ORLIB instances.

There are two main reasons for the faster behavior of DM-HH. First, the computational effort of the adapted construction phase is smaller than the original HH construction, since the elements from a pattern are initially fixed in the solution. Then a smaller number of elements must be processed and inserted into the constructed solution. Second, and most important, the use of patterns leads to the construction of better solutions which will be input for the local search. This incurs in less computational effort taken to converge to a local optimal solution. In the next section, we will further investigate and analyze this behavior.

When executed for the 45 instances from the RW class, both HH and DM-HH reached the best known solutions in all 9 runs for 23 instances. For the other 22 instances, they obtained slightly different solutions. In Table 4.2, the results related to the quality of the obtained solutions are shown. The first column presents the class name, the number of customers and the value of p of the working instances, the second one shows the best known value for this instance, the third and fifth columns present the deviation value of the best cost obtained by HH and DM-HH related to the best known value, and the fourth and sixth columns present the deviation value of the average cost obtained by both strategy.

The deviation value is computed as follows:

$$dev = \frac{(HeuristicCost - BestCost)}{BestCost} \times 100, \tag{4.1}$$

where *HeuristicCost* is the (best or average) cost obtained by the heuristic technique and the *BestCost* is the optimal or best known value for the working instance.

Table 4.1: Time of HH and DM-HH for ORLIB instances

	HH		DM-HH		
Name/Cust/p	Time(s)	SD	Time(s)	SD	%
pmed01/100/5	0.62	0.04	**0.61**	0.04	1.08
pmed02/100/10	0.51	0.03	**0.41**	0.02	20.35
pmed03/100/10	0.51	0.02	**0.39**	0.02	23.14
pmed04/100/20	0.42	0.02	**0.34**	0.07	17.65
pmed05/100/33	0.40	0.03	**0.28**	0.02	29.13
pmed06/200/5	2.26	0.15	**2.23**	0.09	1.38
pmed07/200/10	1.53	0.04	**1.08**	0.11	29.36
pmed08/200/20	1.19	0.02	**0.84**	0.05	29.53
pmed09/200/40	1.14	0.02	**0.80**	0.02	30.10
pmed10/200/67	1.11	0.03	**0.79**	0.02	28.41
pmed11/300/5	3.86	0.13	**3.18**	0.12	17.67
pmed12/300/10	3.07	0.12	**2.96**	0.10	3.40
pmed13/300/30	2.11	0.07	**1.46**	0.06	30.89
pmed14/300/60	2.14	0.03	**1.45**	0.04	32.47
pmed15/300/100	2.43	0.04	**1.72**	0.11	29.07
pmed16/400/5	7.89	0.28	**6.76**	0.08	14.28
pmed17/400/10	5.65	0.11	**4.28**	0.08	24.29
pmed18/400/40	3.73	0.28	**2.42**	0.08	35.32
pmed19/400/80	3.68	0.05	**2.49**	0.05	32.39
pmed20/400/133	4.20	0.11	**3.03**	0.05	27.75
pmed21/500/5	11.11	0.34	**10.95**	0.35	1.45
pmed22/500/10	9.02	0.15	**6.97**	0.15	22.70
pmed23/500/50	4.93	0.20	**3.22**	0.18	34.69
pmed24/500/100	5.40	0.08	**3.73**	0.06	30.93
pmed25/500/167	6.54	0.21	**4.76**	0.09	27.22
pmed26/600/5	18.05	0.25	**13.54**	0.47	24.99
pmed27/600/10	14.69	0.45	**11.62**	0.86	20.90
pmed28/600/60	7.01	0.16	**5.26**	0.20	24.96
pmed29/600/120	8.38	0.32	**5.76**	0.16	31.26
pmed30/600/200	10.89	0.32	**7.67**	0.12	29.57
pmed31/700/5	26.32	0.29	**19.76**	0.58	24.92
pmed32/700/10	17.86	0.51	**11.62**	0.34	34.94
pmed33/700/70	10.43	0.26	**6.72**	0.22	35.57
pmed34/700/140	12.44	0.83	**8.26**	0.16	33.60
pmed35/800/5	33.73	0.77	**27.61**	0.51	18.14
pmed36/800/10	24.87	0.47	**19.09**	0.34	23.24
pmed37/800/80	14.54	0.24	**9.18**	0.59	36.86
pmed38/900/5	51.08	0.74	**40.21**	0.52	21.28
pmed39/900/10	28.90	0.53	**19.56**	0.32	32.32
pmed40/900/90	19.79	0.40	**12.80**	0.34	35.32
Average					25.06

Table 4.2: HH and DM-HH for RW instances

		HH		DM-HH	
Class/Cust/p	Best Known	Best	Avg	Best	Avg
RW/100/10	530	0.000	0.000	0.000	0.000
RW/100/20	277	0.000	0.000	0.000	0.000
RW/100/30	213	0.000	0.000	0.000	0.000
RW/100/40	187	0.000	0.000	0.000	0.000
RW/100/50	172	0.000	0.000	0.000	0.000
RW/250/10	3691	0.000	0.000	0.000	0.000
RW/250/25	1360	0.000	**0.065**	0.000	0.131
RW/250/50	713	0.000	0.000	0.000	0.000
RW/250/75	523	0.000	0.000	0.000	0.000
RW/250/100	444	0.000	0.000	0.000	0.000
RW/250/125	411	0.000	0.000	0.000	0.000
RW/500/10	16108	0.000	0.000	0.000	0.000
RW/500/25	5681	0.000	**0.016**	0.000	0.086
RW/500/50	2626	0.000	**0.131**	0.000	0.161
RW/500/75	1757	0.000	0.000	0.000	0.000
RW/500/100	1379	0.000	**0.056**	0.000	0.064
RW/500/150	1024	0.000	0.000	0.000	0.000
RW/500/200	893	0.000	0.000	0.000	0.000
RW/500/250	833	0.000	0.000	0.000	0.000
RW/1000/10	67811	0.000	**0.000**	0.000	0.107
RW/1000/25	24896	0.108	0.162	**0.000**	**0.096**
RW/1000/50	11274	0.053	0.316	**0.000**	**0.203**
RW/1000/75	7135	0.477	0.662	**0.000**	**0.442**
RW/1000/100	5218	**0.000**	0.290	0.038	**0.200**
RW/1000/200	2704	0.000	**0.033**	0.000	0.049
RW/1000/300	2018	0.000	0.000	0.000	0.000
RW/1000/400	1734	0.000	0.000	0.000	0.000
RW/1000/500	1614	0.000	0.000	0.000	0.000
RW/1500/10	160327	0.000	0.003	0.000	**0.000**
RW/1500/25	59374	0.125	0.246	**0.000**	**0.207**
RW/1500/50	26912	0.305	0.598	**0.000**	**0.469**
RW/1500/75	16921	**0.000**	0.470	0.018	**0.188**
RW/1500/100	12243	0.335	0.573	**0.000**	**0.324**
RW/1500/250	4761	**0.000**	0.173	0.042	**0.142**
RW/1500/500	2867	0.000	0.000	0.000	0.000
RW/1500/750	2422	0.000	0.000	0.000	0.000
RW/2000/10	293073	0.000	0.000	0.000	0.000
RW/2000/25	109481	0.068	0.593	**0.000**	**0.241**
RW/2000/50	50113	**0.000**	0.583	0.010	**0.411**
RW/2000/75	31463	0.372	0.760	**0.000**	**0.527**
RW/2000/100	22514	0.355	0.794	**0.000**	**0.432**
RW/2000/250	8204	0.098	**0.274**	**0.000**	0.288
RW/2000/500	4479	0.022	0.057	**0.000**	**0.042**
RW/2000/750	3560	0.000	0.000	0.000	0.000
RW/2000/1000	3225	0.000	0.000	0.000	0.000
Average		0.051	0.152	**0.002**	**0.107**

Table 4.3: Time of HH and DM-HH for RW instances

	HH		DM-HH		
Class/Cust/p	Time(s)	SD	Time(s)	SD	%
RW/100/10	0.99	0.04	**0.76**	0.04	22.66
RW/100/20	0.62	0.08	**0.42**	0.02	31.29
RW/100/30	0.44	0.02	**0.32**	0.02	28.46
RW/100/40	0.40	0.03	**0.31**	0.02	23.76
RW/100/50	0.33	0.02	**0.28**	0.02	16.94
RW/250/10	7.61	0.28	**6.00**	0.17	21.08
RW/250/25	3.89	0.05	**2.65**	0.05	31.74
RW/250/50	2.33	0.07	**1.57**	0.04	32.35
RW/250/75	1.84	0.04	**1.21**	0.03	34.08
RW/250/100	1.65	0.03	**1.15**	0.04	30.22
RW/250/125	1.40	0.05	**1.06**	0.03	23.89
RW/500/10	35.91	0.67	**30.44**	0.57	15.24
RW/500/25	20.02	0.43	**14.72**	0.94	26.45
RW/500/50	10.26	0.20	**7.16**	0.25	30.21
RW/500/75	7.76	0.12	**5.12**	0.64	34.00
RW/500/100	7.23	0.70	**4.61**	0.16	36.21
RW/500/150	6.60	0.39	**4.08**	0.07	38.14
RW/500/200	6.56	0.25	**4.50**	0.24	31.42
RW/500/250	5.19	0.12	**3.91**	0.15	24.73
RW/1000/10	181.50	35.39	**125.97**	2.90	30.59
RW/1000/25	116.02	19.43	**83.25**	3.48	28.25
RW/1000/50	61.50	0.72	**47.53**	2.20	22.70
RW/1000/75	45.49	1.70	**32.43**	1.35	28.70
RW/1000/100	40.52	1.61	**26.71**	1.12	34.08
RW/1000/200	37.90	0.58	**24.29**	1.29	35.89
RW/1000/300	36.68	0.17	**23.64**	0.33	35.56
RW/1000/400	40.83	2.88	**26.33**	0.24	35.51
RW/1000/500	30.69	0.27	**21.56**	0.29	29.73
RW/1500/10	438.32	7.73	**358.80**	6.90	18.14
RW/1500/25	268.78	7.33	**213.17**	9.10	20.69
RW/1500/50	165.73	3.28	**127.17**	6.46	23.26
RW/1500/75	114.82	0.98	**90.74**	13.65	20.98
RW/1500/100	99.12	10.74	**70.30**	3.15	29.07
RW/1500/250	85.34	0.92	**54.70**	4.39	35.90
RW/1500/500	89.93	0.21	**57.23**	0.33	36.36
RW/1500/750	72.81	0.20	**51.88**	0.26	28.75
RW/2000/10	830.56	35.49	**725.40**	28.33	12.66
RW/2000/25	523.63	11.75	**433.63**	15.92	17.19
RW/2000/50	360.63	12.40	**278.68**	15.15	22.73
RW/2000/75	240.66	5.66	**180.77**	6.36	24.88
RW/2000/100	189.47	4.75	**142.00**	6.59	25.05
RW/2000/250	148.36	11.37	**94.12**	7.86	36.56
RW/2000/500	153.68	3.30	**95.02**	2.26	38.17
RW/2000/750	182.84	22.41	**115.27**	2.39	36.96
RW/2000/1000	132.88	0.29	**92.63**	3.23	30.29
Average					28.26

The smallest values, i.e., the better results, are bold-faced. The last line of these tables presents the average values of each column. Out of the 22 instances for which HH and DM-HH presented different results, in 10 instances, HH reached the best know value in all 9 runs and the DM-HH reached this result in 18 instances (for these instances the Best value is zero). The DM-HH strategy found 11 better results for best values and 4 were found by HH. Considering the average results, DM-HH found 15 better values and HH found 7. These results show that the DM-HH strategy was able to improve slightly the results obtained by HH for the RW class.

In terms of computational time, we can observe in Table 4.3 that, again, the DM-HH strategy was faster than HH. On average, DM-HH was 28.26% faster.

Table 4.4 presents the results related to the quality of the solutions obtained by HH and DM-HH when evaluated for the 18 instances from the FL1400 class. Both strategy reached the best known solutions in all 9 runs for just 3 instances. For the other 15 instances, they obtained slightly different solutions. The HH strategy found 6 better results for best values and just one was found by DM-HH. Considering the average results, HH found 7 better values and DM-HH found 4. Differently from ORLIB and RW classes, these results show that the HH strategy, for the FL1400 class, was able to obtain slightly better results than DM-HH.

However, for the time analysis, DM-HH always improved the HH performance. In Table 4.5, we observe that, once more, the DM-HH strategy was faster than HH. At this time, DM-HH was 30.03% faster.

Table 4.4: HH and DM-HH for FL1400 instances

		HH		DM-HH	
Class/Cust/p	Best Known	Best	Avg	Best	Avg
FL/1400/10	101249.47	0.000	0.000	0.000	0.000
FL/1400/20	57857.55	0.001	0.001	0.001	0.001
FL/1400/30	44013.48	0.000	0.000	0.000	0.000
FL/1400/40	35002.52	0.000	0.000	0.000	0.000
FL/1400/50	29089.78	0.002	0.002	0.002	0.002
FL/1400/60	25161.12	0.000	0.002	0.000	**0.000**
FL/1400/70	22125.53	0.002	0.002	0.002	0.002
FL/1400/80	19870.85	0.000	**0.001**	0.000	0.012
FL/1400/90	17987.94	0.004	0.004	0.004	0.004
FL/1400/100	16551.2	0.006	0.019	0.006	**0.014**
FL/1400/150	12026.43	**0.000**	0.029	0.000	**0.018**
FL/1400/200	9357.74	0.011	**0.028**	**0.000**	0.028
FL/1400/250	7739.8	**0.000**	**0.032**	0.023	0.057
FL/1400/300	6620.92	**0.001**	**0.048**	0.014	0.067
FL/1400/350	5720.88	**0.000**	**0.048**	0.019	0.076
FL/1400/400	5006.75	0.000	0.013	0.000	**0.009**
FL/1400/450	4468.31	**0.000**	**0.062**	0.037	0.095
FL/1400/500	4046.16	**0.000**	**0.039**	0.001	0.049
Average		**0.001**	**0.018**	0.006	0.024

Table 4.5: Time of HH and DM-HH for FL1400 instances

	HH		DM-HH		
Class/Cust/p	Time(s)	SD	Time(s)	SD	%
FL/1400/10	155.05	5.51	**120.78**	3.78	22.10
FL/1400/20	103.13	3.41	**70.79**	1.97	31.36
FL/1400/30	105.19	2.67	**65.67**	1.42	37.57
FL/1400/40	93.92	2.17	**64.76**	1.99	31.05
FL/1400/50	77.66	1.29	**49.60**	1.04	36.13
FL/1400/60	76.28	1.37	**54.31**	2.17	28.80
FL/1400/70	65.84	1.20	**40.97**	1.00	37.78
FL/1400/80	66.35	1.85	**42.61**	1.45	35.77
FL/1400/90	61.96	2.05	**38.34**	1.53	38.13
FL/1400/100	62.62	1.87	**46.11**	0.61	26.36
FL/1400/150	63.00	1.35	**51.59**	1.69	18.12
FL/1400/200	61.79	0.78	**43.41**	3.19	29.75
FL/1400/250	70.98	2.24	**47.72**	3.17	32.78
FL/1400/300	75.40	2.35	**52.50**	2.71	30.37
FL/1400/350	77.13	2.30	**57.20**	3.49	25.83
FL/1400/400	77.62	1.81	**54.95**	3.87	29.20
FL/1400/450	84.72	2.39	**65.67**	2.17	22.48
FL/1400/500	93.48	5.76	**68.27**	3.07	26.97
Average					30.03

5 Strategies Behavior Analysis

In this section, we present some additional analysis of computational experiments performed to illustrate the behavior of both strategies.

Figures 5.1 and 5.2 present the behavior of the construction, local search and path-relinking phases, in terms of the cost values obtained, by HH and DM-HH through the execution of 500 iterations, for the specific instance rw1000-p25.

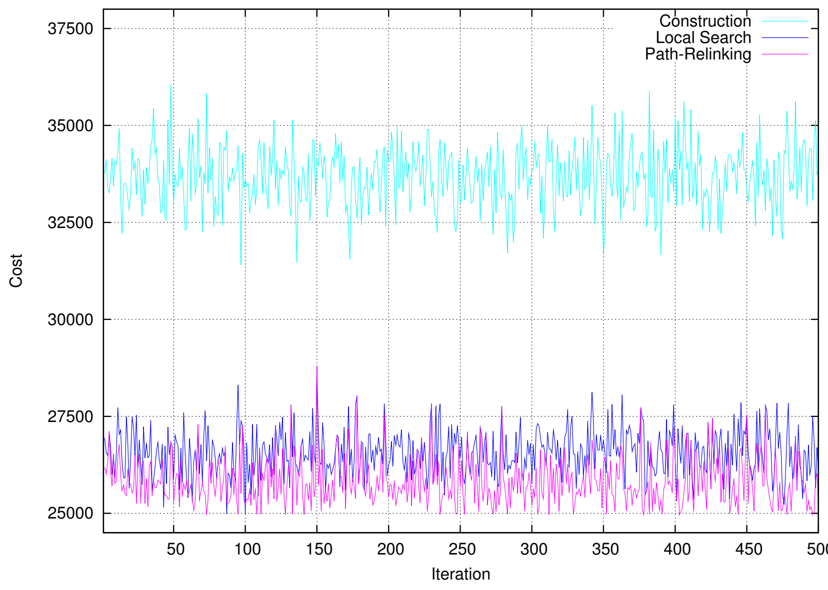

Figure 5.1: One execution of HH for rw1000-p25

These figures show that the local search always reduces the cost of the solution obtained by the construction phase and the path-relinking phase is able to decrease the cost of the solutions obtained by the local search phase in both heuristics HH and DM-HH.

In Figure 5.1, we observe that the behavior of the construction, local search and path-relinking phases performed in HH looks the same through all iterations.

Figure 5.2 shows that the DM-HH strategy provides an improvement in the quality of the solutions reached by the construction, local search and path-relinking phases after iteration 250, where DM-HH starts to use the patterns found by the data mining procedure.

Table 5.1 shows the average costs of the solutions obtained by the construction

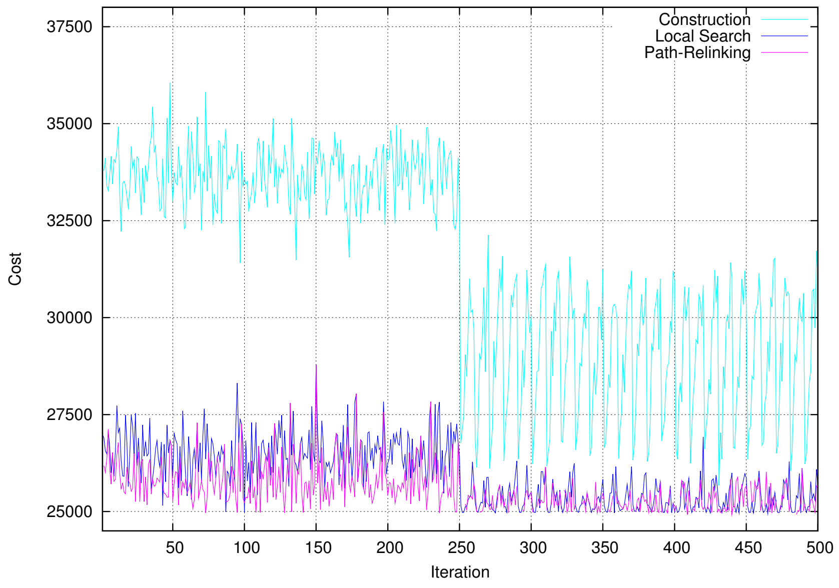

Figure 5.2: One execution of DM-HH for rw1000-p25

phase, and after the local search and path-relinking phases in the first 250 iterations and in the last 250 iterations, i.e., before using the patterns generated by the data mining procedure and using these patterns. All phases present better cost values after using the patterns generated by the data mining procedure. From this decreasing values, we can deduce the benefit from executing the data mining procedure not only for the construction phase but also for the local search and path-relinking phases.

Table 5.1: Average cost values

	Construction	Local search	Path-relinking
First 250 iterations	33675.70	26555.27	25846.06
Next 250 iterations	29007.49	25352.95	25276.90

Figures 5.3 and 5.4 show the behavior of the construction, local search and path-relinking phases, for both strategies HH and DM-HH in terms of the computational time, through the execution of 500 iterations, for the same instance rw1000-p25.

Table 5.2 presents the average computational times used to execute the construction, local search and path-relinking phases in the first 250 iterations and in the last 250 iterations. We can clearly see that computational times of all phases dropped substantially after starting to use the patterns generated by the data mining procedure. The construction phase demands less computational time because it starts from a solution

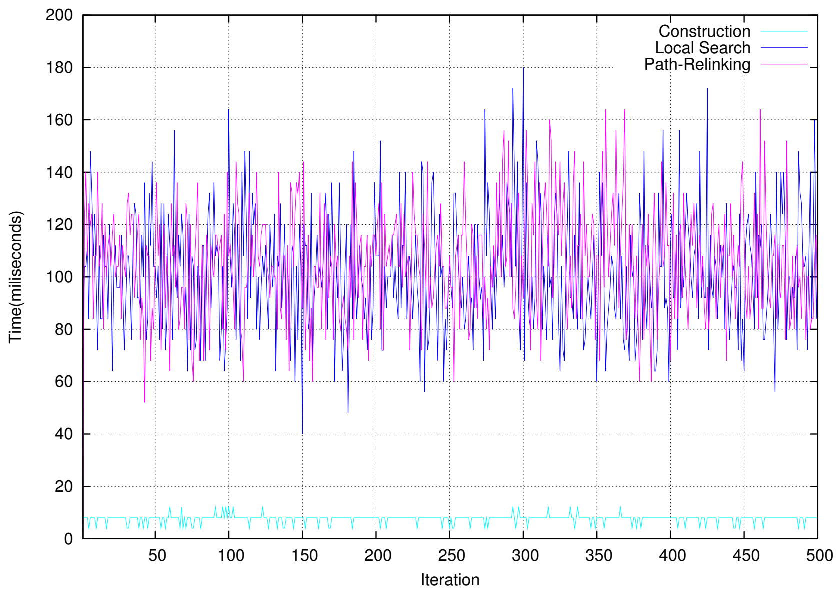

Figure 5.3: One execution of HH for rw1000-p25

partially built using the obtained patterns. The necessary effort required by the local search procedure to find a local optimum decreases due to the better solutions provided by the construction phase. As the solutions generated after the local search procedure present better cost in the iterations which use the data mining patterns, they are more similar to the solutions in the path-relinking pool and the path-relinking procedure takes less time to execute.

Table 5.2: Average computational times

	Construction	Local search	Path-relinking
First 250 iterations	7.66	101.82	105.45
Next 250 iterations	3.95	51.95	82.83

Another experiment was performed to evaluate the time required for HH and DM-HH to achieve a target solution value. Each strategy was run 100 times (with different random seeds), until a target solution was reached for a specific instance. In each run, if the target value was not found in 500 iterations, then the pos-optimization was performed until the target value was found or the elite set used for the pos-optimization procedure was not updated.

The instance rw1000-p25 was used as the test case, and three targets were analyzed: an easy target (value 24964), an intermediate (value 24948), and a more difficult

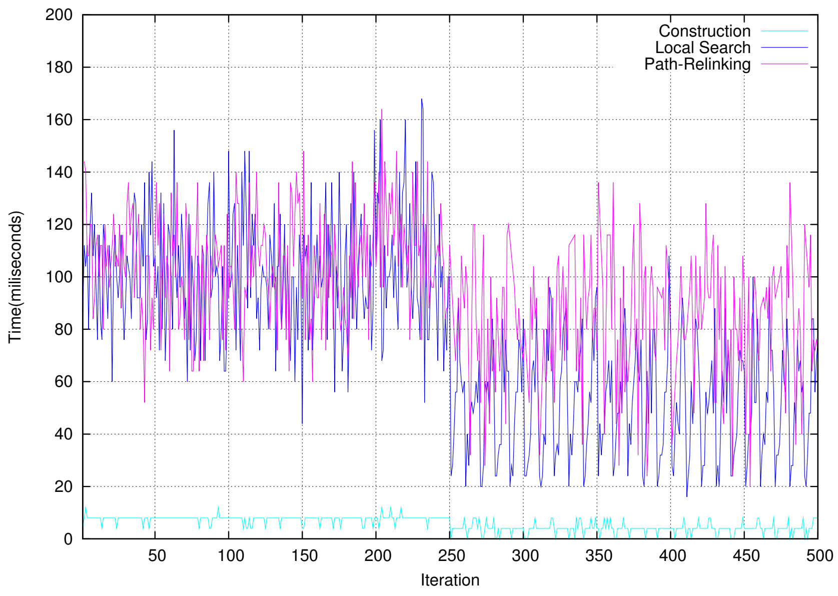

Figure 5.4: One execution of DM-HH for rw1000-p25

one (value 24923).

Figures 5.5, 5.6 and 5.7 show, for each target, the evaluation of the strategies. For each seed, the time in which the target was reached is plotted. DM-HH was able to find the easy target in all executions, the intermediate target in all except one execution and the difficult target in all except 4 executions, while HH was not able to find the easy target in 3 executions, the intermediate target in 23 executions and the difficult target in 33 executions.

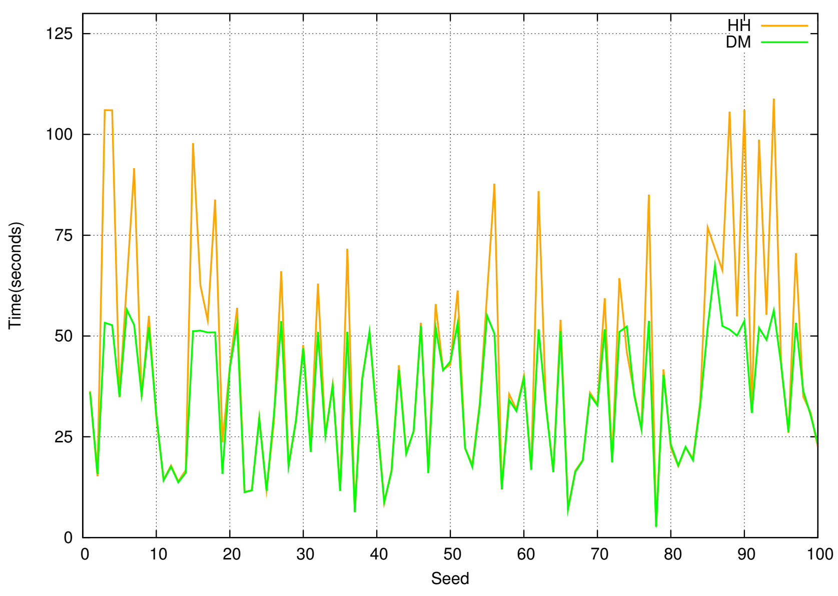

Figure 5.5: Analysis of convergence to an easy target for instance rw1000-p25.

We can observe that in almost all executions, for the three targets, the DM-HH

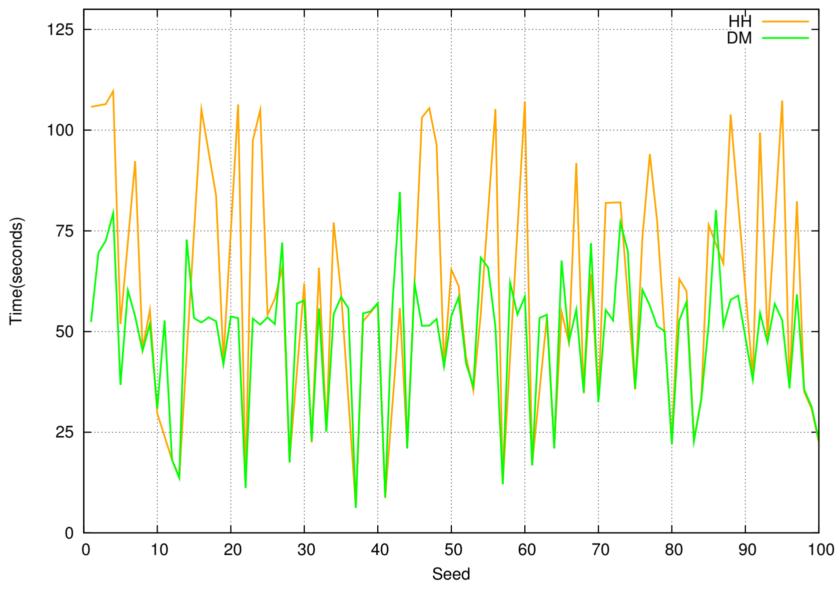

Figure 5.6: Analysis of convergence to an intermediate target for instance rw1000-p25.

reached the target before the HH. Also, DM-HH was able to find the targets more often than HH.

The results of these experiments evidenced that the incorporation of the data-mining procedure into the original HH heuristic was able to improve substantially its efficiency, in terms of the time required to achieve a target solution.

Figures 5.8, 5.9 and 5.10 show another comparison between HH and DM-H strategies, based on *Time-to-target* (TTT) plots [2], which are used to analyze the behavior of randomized algorithms.

A TTT plot is generated, initially, by executing an algorithm several times and measuring the time required to reach a solution at least as good as a target solution. We executed each strategy a hundred times. Then, the i-th sorted running time t_i is associated with a probability $p_i = (i - 1/2)/100$ and the points $z_i = (t_i, p_i)$, for $i = 1, ..., 100$ are plotted. Each plotted point indicates the probability (vertical axis) for the strategy to achieve the target solution in the indicated time (horizontal axis).

The plots presented in Figures 5.8, 5.9 and 5.10 were generated by the execution of HH and DM-HH, for instance rw1000-p25, using the same three target solution values used in the previous experiment.

For the easy target, we observe in Figure 5.8 that HH and DM-HH present similar behaviors until about 50 seconds when the probability for DM-HH to find the

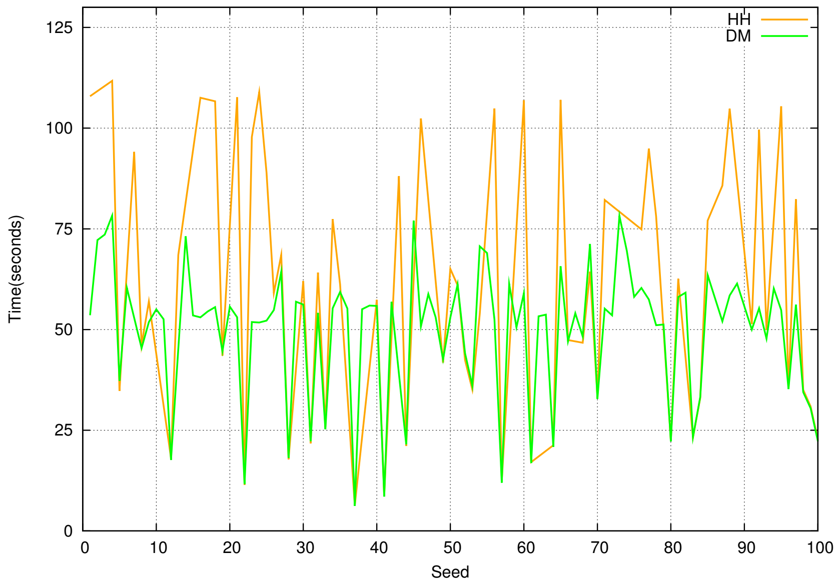

Figure 5.7: Analysis of convergence to a difficult target for instance rw1000-p25.

target value starts to be greater than for HH. This happens because, until the data mining procedure is executed in DM-HH, both strategies obtain the same solution in each iteration, but DM-HH starts to find the target value faster when the patterns are used.

For both intermediate and difficult targets, Figures 5.9 and 5.10 respectively, we observe that DM-HH behaves better than HH. These plots indicate that DM-HH is able to reach difficult solutions faster than HH.

The analysis performed in this section shows that the new data mining version of the hybrid heuristic was able to reach good quality solutions much faster than the original strategy. It also demonstrates that a sophisticated heuristic like HH, which is improved with a memory-based intensification mechanism, like the path-relinking technique, can benefit from the incorporation of a data mining procedure.

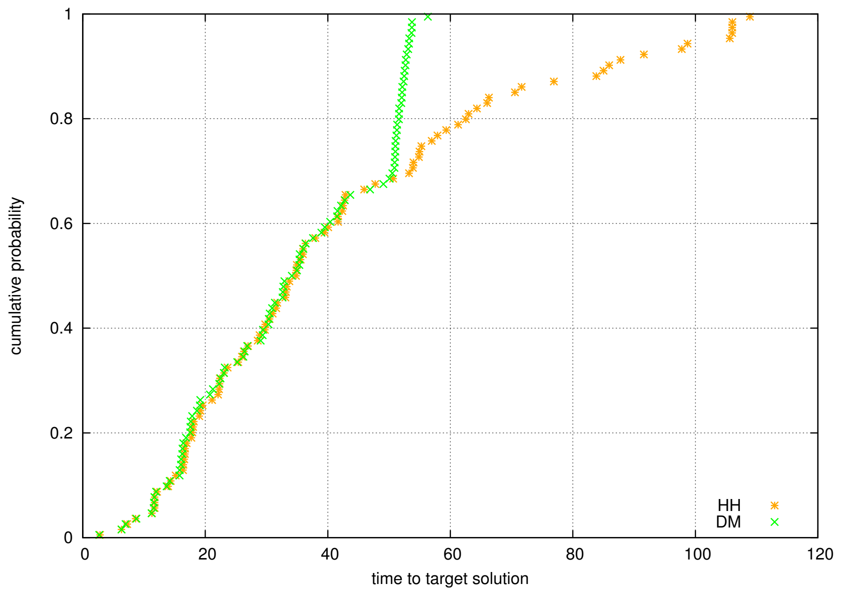

Figure 5.8: Time-to-target plot for an easy target.

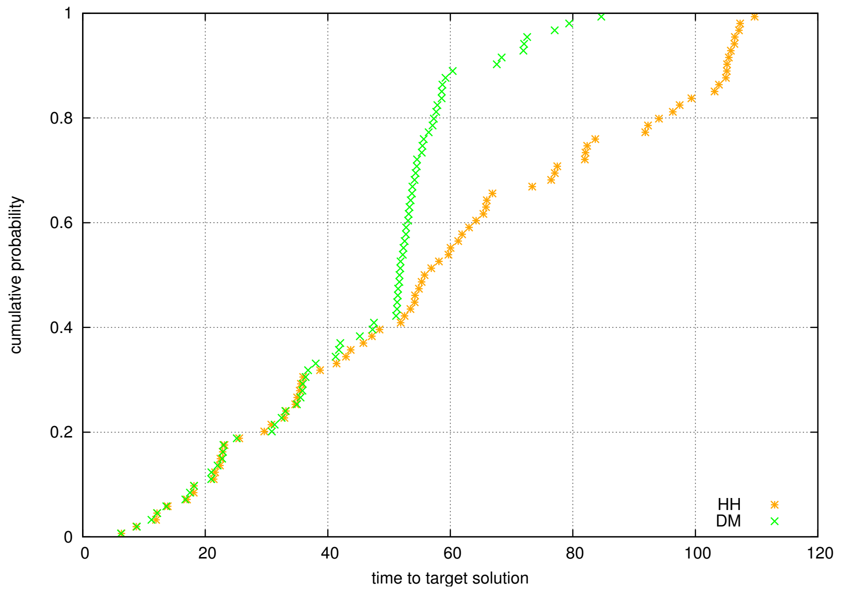

Figure 5.9: Time-to-target plot for an intermediate target.

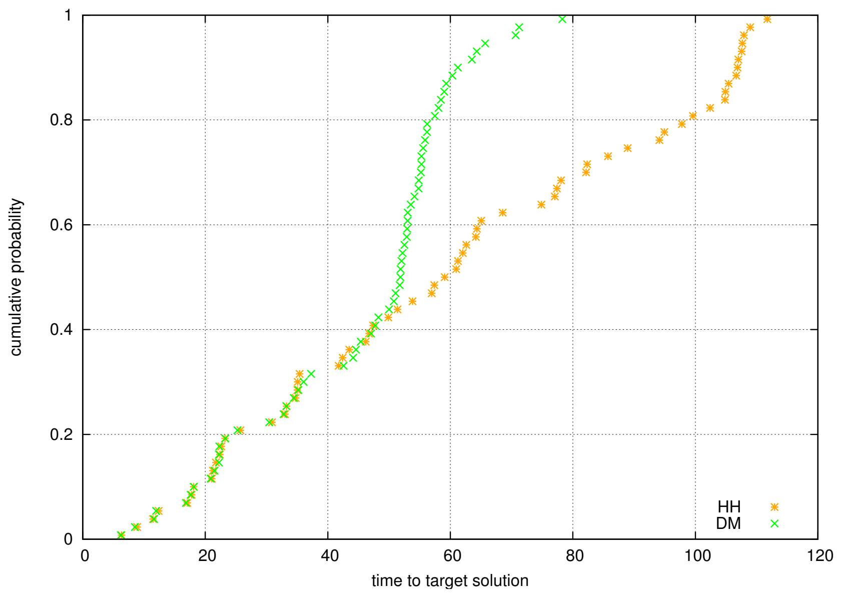

Figure 5.10: Time-to-target plot for a difficult target.

6 Conclusions

In previous works, the DM-GRASP strategy, a hybrid version of the GRASP metaheuristic which incorporates a data mining procedures, was proposed and developed. DM-GRASP has been successfully applied to solve different combinatorial optimization problems. This proposal was based on the hypothesis that patterns extracted from sub-optimal obtained solutions could guide the search for better ones in less computational time.

The aim of this work was to introduce a data mining procedure into a state-of-the-art heuristic for a specific problem in order to give evidences that, when a technique is able to reach the optimal solution, or a near-optimal solution with little chance of improvements, the mined patterns could be used to guide the search for the optimal or near optimal solutions in less computational time.

We then developed the DM-HH, a data mining version of a hybrid and state-of-the-art multistart heuristic to solve the p-median problem. Computational experiments, conducted on a set of instances from the literature, showed that the new version of the hybrid heuristic was able to reach optimal and near-optimal solutions, on average, 27.32% faster than the original strategy, which represents significant savings on execution times.

A secondary contribution of this work was to show that not only the traditional GRASP metaheuristic but also other more sophisticated heuristic, improved with a memory-based intensification mechanism, like the path-relinking technique, could benefit from the incorporation of a data mining procedure.

These encouraging results motivate us, as future work, to try to introduce into other metaheuristics, like tabu search and genetic algorithms, the idea of extracting patterns from sub-optimal solutions using data mining techniques and exploring them in search procedures. We believe that other metaheuristics and many combinatorial optimization problems can benefit from the incorporation of data mining techniques.

References

[1] R. Agrawal and R. Srikant, *Fast algorithms for mining association rules*, Proceedings of the Very Large Data Bases Conference, pp. 487–499 (1994).

[2] R. Aiex, M. G. C. Resende, and C. Ribeiro, *TTT plots: a perl program to create time-to-target plots*, Optimization Letters 4, pp. 355–366 (2007)

[3] J. E. Beasley, *A note on solving large p-median problems*, European Journal of Operational Research 21, pp. 270–273 (1985).

[4] G. Cornuéjols, M. L. Fisher, and G. L. Nemhauser, *Location of Bank Accounts to Optimize Float: An Analytical Study of Exact and Approximate Algorithms*, Management Science 23, pp. 789–810 (1977).

[5] T. A. Feo and M. G. C. Resende, *A probabilistic heuristic for a computationally difficult set covering problem*, Operations Research Letters 8, pp. 67–71 (1989).

[6] T. A. Feo and M. G. C. Resende, *Greedy randomized adaptive search procedures*, Journal of Global Optimization 6, pp. 109–133 (1995).

[7] P. Festa and M. G. C. Resende, *An annotated bibliography of GRASP - Part I: Algorithms*, International Transactions in Operational Research 16, pp. 1–24 (2009).

[8] P. Festa and M. G. C. Resende, *An annotated bibliography of GRASP - Part II: Applications*, International Transactions in Operational Research 16, pp. 131–172 (2009).

[9] C. Fleurent and F. Glover, *Improved Constructive Multistart Strategies for the Quadratic Assignment Problem Using Adaptive Memory*, INFORMS J. on Computing 2, pp. 198-204 (1999).

[10] F. Glover, *Multi-start and strategic oscillation methods – Principles to exploit adaptive memory*, Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, Kluwer, pp. 1–24 (2000).

[11] F. Glover, M. Laguna, and R. Martí, *Fundamentals of scatter search and path-relinking* Control and Cybernetics 19, pp. 653–684 (1977).

[12] F. Glover, M. Laguna, and R. Martí, *Scatter search and path relinking: Advances and applications*, Handbook of Metaheuristics, Kluwer, pp. 1–35 (2003).

[13] B. Goethals and M. J. Zaki, *Advances in Frequent Itemset Mining Implementations: Introduction to FIMI-03*, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (2003).

[14] G. Grahne and J. Zhu, *Efficiently using prefix-trees in mining frequent item-sets*, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (2003).

[15] J. Han, J. Pei, and Y. Yin, *Mining frequent patterns without candidate generation*, Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 1–12 (2000).

[16] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2^{nd} Ed., Morgan Kaufmann Publishers (2006).

[17] P. Hansen and N. Mladenović, *Variable Neighborhood Search for the p-Median*, Location Science 5, pp. 207–226 (1997).

[18] P. Hansen, N. Mladenović, and D. Perez-Brito, *Variable Neighborhood Decomposition Search*, Journal of Heuristics 7, pp. 335–350 (2001).

[19] O. Kariv and L. Hakimi, *An algorithmic approach to network location problems, Part II: The p-medians*, SIAM Journal of Applied Mathematics 37, pp. 539–560 (1979).

[20] S. Lin and B.W. Kernighan, *An effective heuristic algorithm for the traveling salesman problem*, Operations Research 21, pp. 498–516 (1973).

[21] A. Lodi, K. Allemand, and T. M. Liebling, *An evolutionary heuristic for quadratic 0-1 programming*, European Journal of Operational Research 119, pp. 662–670 (1999).

[22] S. Orlando, P. Palmerimi, and R. Perego, *Adaptive and resource-aware mining of frequent sets*, Proceedings of the IEEE International Conference on Data Mining, pp. 338–345 (2002).

[23] I. Osman and G. Laporte, *Metaheuristics: A bibliography*, Annals of Operations Research 63, pp. 513–623 (1996).

[24] M.R. Rao, *Cluster analysis and mathematical programming*, Journal of the American Statistical Association 66, pp. 622–626 (1971).

[25] G. Reinelt, *TSPLIB: A traveling salesman problem library*, ORSA Journal on Computing 3, pp. 376–384 (1991), *http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.*

[26] M. G. C. Resende and R. F. Werneck, *A hybrid heuristic for the p-median problem*, Journal of Heuristics 10, pp. 59–88 (2004).

[27] M. G. C. Resende and R .F . Werneck, *On the implementation of a swap-based local search procedure for the p-median problem*, Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments – ALENEX03, pp. 119–127 (2003).

[28] M. H. F. Ribeiro, V. F. Trindade, A. Plastino, and S. L. Martins, *Hybridization of GRASP metaheuristic with data mining techniques*, Proceedings of the ECAI Workshop on Hybrid Metaheuristics, pp. 69–78 (2004).

[29] M. H. F. Ribeiro, V. F. Trindade, A. Plastino, and S. L. Martins, *Hybridization of GRASP metaheuristic with data mining techniques*, Journal of Mathematical Modeling and Algorithms 5, pp. 23–41 (2006).

[30] S. Salhi, *Heuristic Search: The Science of Tomorrow*, OR48 Keynote Papers, Operational Research Society, pp. 38–58 (2006).

[31] L. F. Santos, M. H. F. Ribeiro, A. Plastino, and S. L. Martins, *A hybrid GRASP with data mining for the maximum diversity problem*, Proceedings of the International Workshop on Hybrid Metaheuristics, LNCS 3636, pp. 116–127 (2005).

[32] L. F. Santos, C. V. Albuquerque, S. L. Martins, and A. Plastino, *A hybrid GRASP with data mining for efficient server replication for reliable multicast*, Proceedings of the IEEE GLOBECOM Conference (2006).

[33] L. F. Santos, S. L. Martins, and A. Plastino, *Applications of the DM-GRASP heuristic: A survey*, International Transactions in Operational Research 15, pp. 387–416 (2008).

[34] E. L. F. Senne and L. A. N. Lorena, *Langrangean/Surrogate Heuristics for p-Median Problems*, Computing Tools for Modeling, Optimization and Simulation: Interfaces in

Computer Science and Operations Research, M. Laguna and J. L. González-Velarde (eds.), Kluwer, pp. 115–130 (2000).

[35] E. D. Taillard, *Heuristic Methods for Large Centroid Clustering Problems*, Journal of Heuristics 9, pp. 51–74 (2003).

[36] E. G. Talbi, *A taxonomy of hybrid metaheuristics*, Journal of Heuristics 8, pp. 541–564 (2002).

[37] B. C. Tansel, R. L. Francis, and T. J. Lowe. *Location on networks: A survey*, Management Science 29, pp. 482–511 (1983).

[38] M. B. Teitz and P. Bart, *Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph*, Operations Research 16, pp. 955–961 (1968).

[39] H.D. Vinod, *Integer programming and the theory of groups*, Journal of the American Statistical Association 64, pp. 506–519 (1969).

[40] R. Whitaker, *A Fast Algorithm for the Greedy Interchange of Large-Scale Clustering and Median Location Problems*, INFOR 21, pp. 95–108 (1983).

[41] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques.* 2^{nd} Ed., Morgan Kaufmann Publishers (2005).