DOI: 10.15514/ISPRAS-2019-31(2)-2

Virtual Savant for the Knapsack Problem: learning for automatic resource allocation

[1,2] R. Massobrio, ORCID: 0000-0002-0040-3681 <renzom@fing.edu.uy>
[1] B. Dorronsoro Díaz, ORCID: 0000-0003-0481-790X <bernabe.dorronsoro@uca.es>
[2] S.E. Nesmachnow Cánovas, ORCID: 0000-0002-8146-4012 <sergion@fing.edu.uy>
[1] Universidad de Cádiz, Av. de la Universidad, 10, 11519, Cádiz, España
[2] Universidad de la República, Herrera y Reissig 565, 11300, Montevideo, Uruguay.

Abstract. This article presents the application of Virtual Savant to solve resource allocation problems, a widely-studied area with several real-world applications. Virtual Savant is a novel soft computing method that uses machine learning techniques to compute solutions to a given optimization problem. Virtual Savant aims at learning how to solve a given problem from the solutions computed by a reference algorithm, and its design allows taking advantage of modern parallel computing infrastructures. The proposed approach is evaluated to solve the Knapsack Problem, which models different variant of resource allocation problems, considering a set of instances with varying size and difficulty. The experimental analysis is performed on an Intel Xeon Phi many-core server. Results indicate that Virtual Savant is able to compute accurate solutions while showing good scalability properties when increasing the number of computing resources used.

Keywords: virtual savant; machine learning; parallel computing; resource allocation; knapsack problem; many-core.

For citation: Massobrio R., Dorronsoro Díaz B., Nesmachnow Cánovas S.E. Virtual Savant for the Knapsack Problem: learning for automatic resource allocation. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 2, 2019. pp. 21-32. DOI: 10.15514/ISPRAS-2019-31(2)-2

Acknowledgements. The work of Renzo Massobrio and Sergio Nesmachnow is partly supported by ANII and PEDECIBA, Uruguay. The work of Renzo Massobrio is partly supported by Fundación Carolina, Spain. Bernabé Dorronsoro acknowledges the Spanish MINECO and ERDF for the support provided under contracts TIN2014-60844-R (the SAVANT project) and RYC-2013-13355.

Виртуальный Эрудит для решения задачи о рюкзаке: обучение автоматическому распределению ресурсов

[1,2] Р. Массобрио, ORCID: 0000-0002-0040-3681 <renzom@fing.edu.uy>
[2] Б. Дорронзоро Диаз, ORCID: 0000-0003-0481-790X <bernabe.dorronsoro@uca.es>
[1] С.Е. Несмачнов Кановас, ORCID: 0000-0002-8146-4012 <sergion@fing.edu.uy>
[1] Республиканский университет,
Уругвай, 11300, Монтевидео, ул. Хулио Эррера-и-Рейссиг, 565
[2] Кадисский университет,
Испания, 11001, Кадис, ул. Анча, 16

Аннотация. В этой статье представлено применение метода Виртуального Эрудита (Virtual Savant) для решения проблем распределения ресурсов, широко изученной области с несколькими реальными приложениями. Virtual Savant – это новый метод мягких вычислений, в котором используются методы машинного обучения для вычисления решений данной проблемы оптимизации. Цель Virtual Savant – научиться решать данную проблему с помощью решений, рассчитанных по эталонному алгоритму, а

его дизайн позволяет использовать преимущества современных параллельных вычислительных инфраструктур. Предложенный подход оценивается на решении задачи о рюкзаке, которая моделирует различные варианты задач распределения ресурсов, учитывая набор экземпляров разного размера и сложности. Экспериментальный анализ проводился на многоядерном сервере Intel Xeon Phi. Результаты показывают, что Virtual Savant способен вычислять точные решения, демонстрируя хорошие свойства масштабируемости при увеличении объема используемых вычислительных ресурсов.

Ключевые слова: виртуальный эрудит; машинное обучение; параллельная обработка; распределение ресурсов; задача о рюкзаке; многоядерные процессоры

Для цитирования: Массобрио Р., Дорронзоро Диаз Б., Несмачнов Кановас С.Е. Виртуальный Эрудит для решения задачи о рюкзаке: обучение автоматическому распределению ресурсов. Труды ИСП РАН, том 31, вып. 2, 2019 г., стр. 21-32 (на английском языке). DOI: 10.15514/ISPRAS-2019-31(2)-2

Благодарности. Работа Р. Массобрио и С. Несмачнова частично поддерживается ANII и PEDECIBA, Уругвай. Работа Р. Массобрио частично поддерживается Фондом Каролина, Испания. Б. Дорронсоро благодарит испанские фонды MINECO и ERDF за поддержку, предоставляемую по контрактам TIN2014-60844-R (проект SAVANT) и RYC-2013-13355.

1. Introduction

Resource allocation refers to the assignment of a number of available resources or assets to different issues or items. Resource allocation is an important concept that models several situations and problems arising in economics, strategic planning, project management, scheduling, logistics, production, engineering, and many other related areas [1].

Many resource allocation problems are modeled by the general framework formulated by the Knapsack Problem (Knapsack Problem) [2]. Knapsack Problem is a combinatorial optimization problem that, given a set of items with associated weights and profits, proposes determining the number of each item to include in a collection (i.e., the knapsack) in order to maximize the total profit while ensuring that the total weight is less than or equal to a given limit (i.e., the knapsack capacity). Different allocation problems are modeled by considering the capacity of the knapsack as the available amount of a given resource and the items as activities to which the resource can be allocated.

This article describes a generic paradigm that proposes applying a computational intelligence approach to find accurate solutions to resource allocation problems modeled by the 0/1 Knapsack Problem in short computation times. 0/1 Knapsack Problem is a binary version of the Knapsack Problem where each item is considered as an atomic unit, i.e., each item can be included in the knapsack as a unit or discarded (i.e., it cannot be split to fill the knapsack). This binary version of the Knapsack Problem allows modeling interesting resource allocation problems such as activities in project management, scheduling and location problems, feature selection, among others.

The Virtual Savant paradigm is applied to solve the 0/1 Knapsack Problem, which models allocation problems. Virtual Savant is a novel method that uses machine learning techniques to learn how a reference algorithm solves a given problem [3]. Virtual Savant is inspired by the savant syndrome, a rare condition in which a human demonstrates mnemonic or computing abilities far superior to what would be considered normal. As an example, some patients with savant syndrome (*savants*) are able to enumerate and identify huge prime numbers without the underlying knowledge of what a prime number is, or accurately determine the day of the week of a given date extremely fast. Reported evidence suggests that patients with savant syndrome use pattern recognition in order to efficiently solve problems [4,5,6].

The Virtual Savant paradigm proposes applying a learning approach using computational intelligence to predict the results computed by a reference algorithm that solves a given problem [7,3]. Virtual Savant receives as input a set of problem instances and the results computed by the

reference algorithm, which is used to train a machine learning classifier. Once the training phase is completed, Virtual Savant can be applied to solve new, unknown, and even larger problem instances.

In this way, the Virtual Savant paradigm aims at learning the behavior of a given resolution algorithm in order to generate a completely different program that reproduces an analogous but unknown process to compute accurate results for the same problem. Furthermore, the resulting generated program is lightweight and can take advantage of modern massively parallel computing architectures to provide a fast and powerful problem solving schema.

Following previous works [8,9], this article describes a deeper study on how to solve the 0/1 Knapsack Problem using Virtual Savant. The first evaluation of Virtual Savant in a parallel environment (Intel Xeon Phi 7250 server) to solve the 0/1 Knapsack Problem is presented. The accuracy of the proposed approach is studied as well as its parallel capabilities and performance on a many-core computing environment. Experimental results when solving 0/1 Knapsack Problem instances of varying size and difficulty suggest that the proposed approach is able to compute competitive solutions while showing good scalability properties when increasing the number of processing elements.

The article is organized as follows. Section 2 presents the 0/1 Knapsack Problem formulation, introduces Virtual Savant and presents an overview of the related literature. Section 3 outlines the application of Virtual Savant to the 0/1 Knapsack Problem. Section 4 presents the experimental evaluation of the proposed approach and, finally, Section 5 presents the conclusions and main lines of future work.

2. Problem and method

This section introduces the 0/1 knapsack problem, describes the Virtual Savant paradigm, and presents a review of the related literature.

2.1 0/1 Knapsack Problem formulation

The 0/1 Knapsack Problem is a classic combinatorial optimization problem which is proven to be *NP*-hard [10]. The mathematical formulation is as follows. Given a set I of items, each with a profit p_i and a weight w_i, the 0/1 Knapsack Problem consists in finding a subset of items that maximizes the total profit, without exceeding the weight capacity W of the knapsack.

Eq. 1 shows the problem formulation, where $x_i \in \{0,1\}$ indicates whether item i is included or not in the knapsack.

$$argmax\ \{\sum_{i=1}^{n} p_i x_i \mid \sum_{i=1}^{n} w_i x_i \leq W\} \qquad (1)$$

Despite its straightforward formulation, the 0/1 Knapsack Problem has a large solution space and is frequently used as a benchmark to evaluate optimization algorithms. Additionally, the 0/1 Knapsack Problem can be used to model several optimization problems with direct real-world applications in many fields.

In the context of this work, the 0/1 Knapsack Problem is useful to evaluate the Virtual Savant paradigm for several reasons: i) it is a *NP*-hard optimization problem; ii) it allows studying the behavior of Virtual Savant in problems with binary variables and simple constraints; iii) a large dataset of problem instances is publicly available with varying size and difficulty.

2.2 Virtual Savant

Virtual Savant is a novel paradigm to automatically generate programs that solve optimization problems in a massively parallel fashion [11]. The paradigm is inspired by the savant syndrome, a rare condition in which a person with significant mental disabilities has certain abilities far in excess of what would be considered normal [5]. People with this condition (*savants*) usually excel at one specific skill such as art, memory, rapid calculation, or musical abilities. The methods used by savants to solve problems are not fully understood due to the difficulties in communicating with

them, since the syndrome is usually associated with autism. The main hypothesis states that savants learn through pattern recognition [4]. This mechanism allows savants to solve a given problem without understanding the underlying principles (e.g., being able to enumerate prime numbers without understanding what a prime number is).

In analogy to the savant syndrome, Virtual Savant consists in training a machine learning classifier to automatically learn how to solve an optimization problem from a set of observations, which are usually obtained from a reference algorithm that solves the same problem. Once the training phase is completed, Virtual Savant can emulate the reference algorithm to solve new, unknown, and even larger problem instances, without the need of any further training. The Virtual Savant paradigm consists of two phases: *classification*, where results for unknown problem instances are predicted, and *improvement*, where predicted results are further improved using specific search procedures.

2.3 Related work

The 0/1 Knapsack problem has been widely studied in the operations research field. Nemhauser and Ullman [12] presented an exact algorithm to solve the 0/1 Knapsack Problem based on dynamic programming. The proposed algorithm was devised to solve capital allocation problems with constrained budgets, in the field of economics. Later, an optimized implementation of the original Nemhauser-Ullman algorithm was proposed by Harman et al. [13]. This version was applied to solve instances of the Next Release Problem, an optimization problem from software engineering where the goal is to determine the features to include in a new release of a given software product [14]. The optimized implementation by Harman et al. is used in our work to train the proposed Virtual Savant for 0/1 Knapsack Problem.

Few articles were found in the related literature applying machine learning techniques to solve optimization problems, in line with the Virtual Savant proposal.

Vinyals et al. [15] introduced Pointer Networks (*ptr-nets*), a model based on recurrent neural networks. Similarly to the approach applied in Virtual Savant, *ptr-nets* are trained by observing solved instances of a given problem and the proposed scheme is also able to deal with variable size outputs. The proposed model was applied to solve three different discrete combinatorial optimization problems: finding planar convex hulls, computing Delaunay triangulations, and solving the planar Travelling Salesman Problem. Experimental results indicated that the trained models were able to address problem instances larger than those seen during training and find competitive results for the studied problems.

More recently, Hu et al. [16] applied a similar approach to the one proposed by Vinyals et al. to the three-dimensional bin packing problem, a specific variant of an allocation problem. A deep reinforcement learning approach is used to decide the sequence to pack items in a bin, while the empty space and the spatial orientation in which the items are placed inside the bin are calculated by heuristic methods. The reported experimental results showed that the proposed approach outperformed a specific heuristic for the problem. Improvements of 5% on average over the baseline results were obtained for the problem instances studied.

Our previous works were able to obtain promising results when applying Virtual Savant to a task scheduling problem [17,18,7,11]. The application of Virtual Savant to the 0/1 Knapsack Problem has been previously studied in [8,9]. This article extends those two previous works by evaluating the parallel capabilities of the Virtual Savant model in a many-core parallel infrastructure.

3. Virtual Savant for the 0/1 Knapsack Problem

This section describes the application of the Virtual Savant paradigm to the 0/1 Knapsack Problem. The Virtual Savant implementation for the 0/1 Knapsack Problem uses Support Vector Machines (SVMs) for the classification phase. SVMs are trained using Nemhauser-Ullmann as a reference algorithm, which computes exact solutions for the 0/1 Knapsack Problem [13]. Each item of the problem instance is considered individually during the training phase of Virtual Savant. Therefore,

each feature vector holds the weight and profit of the item, along with the capacity of the knapsack. The classification label is 0/1, indicating whether the reference algorithm included (or not) the item in the knapsack. Thus, a single solution of the reference algorithm provides as many observations as the number of items in the instance. The LIBSVM framework with a Radial Basis Function kernel was used [19]. A specific fork of the LIBSVM package was designed to improve training times on many-core architectures [20]. Fig. 1 outlines the training scheme for Virtual Savant to solve the 0/1 Knapsack Problem.

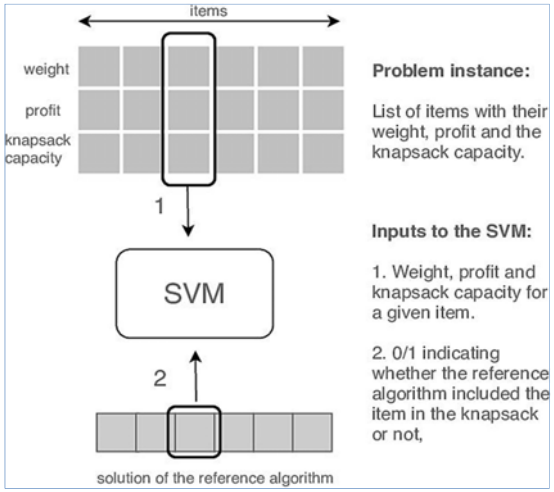

Fig. 1. Training scheme of Virtual Savant for the 0/1 Knapsack Problem

Fig. 2 presents the complete model of Virtual Savant to solve the 0/1 Knapsack Problem. Once the learning process is completed, Virtual Savant uses (in parallel) multiple instances of the trained SVM to predict whether or not to include each item in the knapsack. These decisions are independent for each item, providing Virtual Savant with a high degree of parallelism. The output of the classification phase is a vector that holds, for each item, the probability of including it in the knapsack. Since the length of the training vectors is fixed (3 features + 1 label), there is no need to re-train the SVM to solve problem instances of different size (i.e., with varying number of items). This allows Virtual Savant to easily scale to problem instances of larger dimensions, without requiring any additional training process.

The improvement phase takes as input the resulting vector of probabilities computed in the prediction phase. One candidate solution is generated per computing resource available, by randomly sampling according to the probabilities of including each item. Finally, a local search heuristic is applied over each generated solution. The local search operator considered in this work is very simple, just performing random modifications on the items to include or not. On each step of the local search, a randomly-chosen bit in the solution is flipped, the new solution is evaluated, and the local search continues from that solution if an improvement is made. Algorithm 1 describes the method to evaluate the score of a solution in the local search procedure, considering a solution with profit P, weight W, overweight $O = W - C$, where C is the knapsack capacity; $k > 0$; $m \in (0,1)$. P, W, and O are scaled using the minimum and maximum weight and profit values in the problem instance. The improvement phase, as well as the prediction phase, is massively parallel,

since more local searches can be spawned as more computing resources are available.

Algorithm 1. Score assignment for solutions during the local search

```
input: solution, instance
scale (W, P, O, C, instance)
if O≤0 then
   return P
else if O ≤ m · C then
   return P − k · O
else
   return −O
```

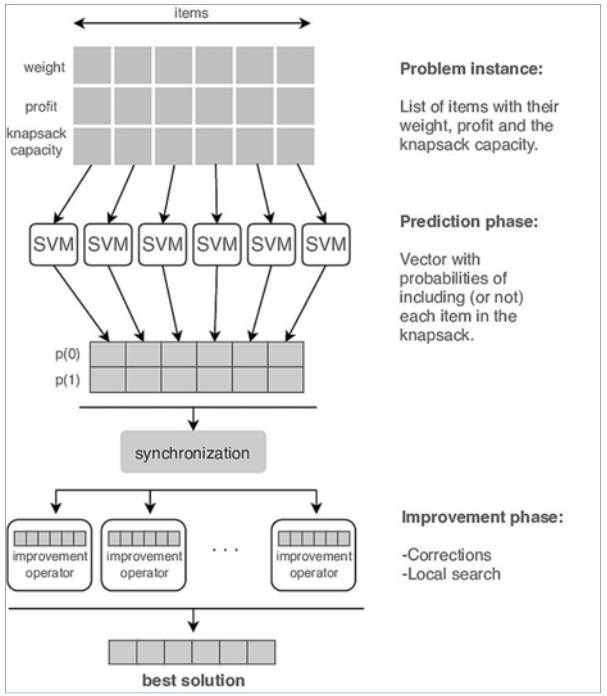

Fig. 2. Model of Virtual Savant applied to the 0/1 Knapsack Problem

Two corrections schemes are included in the improvement phase in order to ensure that the returned solution satisfies the knapsack capacity restriction:

- *Greedy correction by profit (CP)*: iteratively removes the item with lower benefit until the total weight is lower than, or equal to, the knapsack capacity.
- *Greedy correction by weight (CW)*: iteratively searches for the items with weight higher than, or equal to, the overweight of the solution and removes the one with the lowest weight among them. If no item satisfies this condition, it removes the one with the highest weight.

The corrections are applied to each tentative solution after the local search, to ensure that the returned solution satisfies the knapsack capacity constraint. After all local searches and corrections are completed, the overall best solution found is returned.

4. Experimental analysis

This section reports the experimental analysis of the proposed Virtual Savant for the 0/1 Knapsack Problem.

4.1 Problem instances

The evaluation was performed over benchmark problem instances with different size and correlation between weight and profit of items. The correlation is related to the difficulty to solve an instance [13]. The benchmark includes 50 datasets, each with instances of size 100 to 1500 items (stepsize: 100). For each problem size, correlation varies from 0.0 to 1.0 (stepsize: 0.05). The benchmark, including a total of 15.750 problem instances, is publicly available at ucase.uca.es/nrp.

4.2 SVM training

The training phase was performed using dataset 1, to evaluate three different feature configurations. Results show that the best accuracy results were achieved when using item weight, item profit, and knapsack capacity. Regarding the size of the training set, results show that training with 15% of dataset 1 allows achieving good accuracy metrics. Increasing the number of observations results in marginal accuracy improvements, while significantly increasing training times.

The parameters for the SVM (C) and the RBF kernel (γ) were configured prior to the experimental evaluation. Cross-validation was performed over a set of 5.000 samples randomly selected from dataset 1. Results suggest that the best results are computed with C=8192 and γ=0.5. Average accuracy for all datasets increased from 89.35% to 90.48% after parameter configuration. For the improvement phase, the parameters of the score assignment function in the local search were configured to m=0.2 and k=2 and the stopping criterion was set to 1000 iterations.

4.3 Experimental results

After configuration, the trained SVM was used to evaluate the complete Virtual Savant model on datasets 2 to 5. These datasets are completely new for the algorithm, as they were not used during the training phase. The experimental evaluation focused on both the quality of the solutions and the performance and scalability when using a massively parallel computing infrastructure.

4.3.1 Hardware platform

A many-core computing infrastructure was used in the experimental analysis, in order to evaluate the capabilities of Virtual Savant to compute accurate results over a massively parallel platform. A typical many-core computing infrastructure consists of tens or thousands of simpler independent cores. The use of many-core processors has been increasing in the past years, with extensive applications in embedded systems and high-performance computing platforms [21].

Many-core architectures can be programmed using the standard CPU model without needing specific knowledge about the underlying parallel hardware. Even without including platform-specific features, many-core systems offer support for serial legacy code [22]. The evaluation of Virtual Savant for the 0/1 Knapsack Problem was performed on an Intel Xeon Phi 7250 processor with 68 cores and 64GB RAM.

4.3.2 Scalability

Virtual Savant approach is elastic and adapts to the underlying hardware platform: if more computing resources are available, Virtual Savant can use them on both the prediction and the improvement phase. In the prediction phase, the computational load of predicting whether each item is included or not in the knapsack is balanced among the computing resources available. In the

improvement phase, Virtual Savant takes advantage of available resources to execute more local searches on tentative solutions, thus increasing the probability of computing more accurate results.

The scalability of Virtual Savant when using a varying number of computing elements was evaluated for the prediction and improvement phases. Fig. 3 reports the average execution time (in seconds) for all problem instances studied when varying the number of threads.

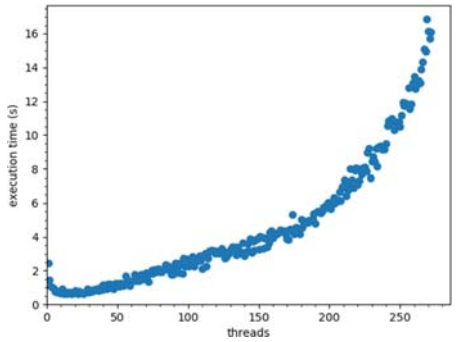

Fig. 3. Execution time varying the number of threads

Results show that Virtual Savant scales very well when increasing the number of threads up to the number of cores available. When more threads are spawned, the performance starts degrading due to threads sharing resources. Consequently, the remainder of the experimental evaluation was performed using 68 threads. These results confirm the good scalability properties of Virtual Savant.

4.3.3 Virtual Savant: prediction phase accuracy

Boxplots in Figs. 4 and 5 correspond to the accuracy achieved during the prediction phase of Virtual Savant grouping problem instances by size and weight/profit correlation, respectively. The median prediction accuracy of the SVM is larger than 90% for all problem sizes studied. No significant differences are noticed among instances of different sizes. On the other hand, significant differences can be observed in the accuracy of the prediction phase on instances with varying weight/profit correlation. Instances with weight/profit correlation of 0.5 are the simplest to predict for the SVM, with a median accuracy value of over 97%. Additionally, in the worst case, the median accuracy of the SVM is larger than 80%.

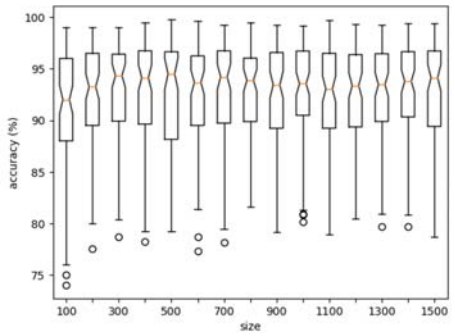

Fig. 4. Prediction accuracy for instances grouped by size.

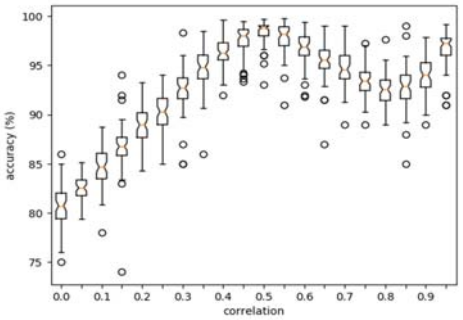

Fig. 5. Prediction accuracy for instances grouped by weight/profit correlation.

4.3.4 Virtual Savant: quality of solutions

Results computed by Virtual Savant were compared with the known optima for the studied instances, to evaluate the efficacy of the proposed approach. Table 1 reports the average ratio to the optima for problem instances grouped by size. Table 2 reports the average ratio to the optimum, grouping instances by the correlation between weight and profit of items.

Results achieved by Virtual Savant grouped by instance size differ from the known optima in just 2-4% on average for all problem instances studied. This is an encouraging result considering that the improvement phase of Virtual Savant consists in a straight-forward local search which does not incorporate any specific knowledge of the problem, thus making it potentially extensible to other related optimization problems. When looking at results grouped by weight/profit correlation, Virtual Savant allows computing accurate results for all problem instances studied. In the worst case, Virtual Savant differs from the optimum in 6% on average (for instances with no correlation between weight and profit).

Table 1. Average ratio to optimum with varying size

size	100	200	300	400	500	600	700	800
ratio	0.98	0.98	0.98	0.97	0.97	0.97	0.97	0.97
size	900	1000	1100	1200	1300	1400	1500	
ratio	0.97	0.97	0.97	0.97	0.97	0.97	0.96	

Table 2. Average ratio to optimum with varying correlation

correlation	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
ratio	0.94	0.95	0.96	0.97	0.97	0.98	0.98	0.99	0.99	0.99
correlation	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
ratio	0.99	0.98	0.97	0.97	0.96	0.96	0.96	0.96	0.97	0.98

5. Conclusions and future work

This article presents the application of Virtual Savant to the 0/1 Knapsack Problem. Virtual Savant learns from a reference algorithm in order to generate a new program that can solve the same optimization problem in a massively parallel fashion. Experimental results show that Virtual Savant allows computing competitive results in reduced execution times thanks to its scalability when using multiple computing elements. The experimental analysis, performed on a many-core infrastructure, showed the good scalability properties of the Virtual Savant paradigm and its elasticity to adapt to modern massively-parallel computing infrastructures.

The main lines of future work include applying other machine learning classifiers, using different

heuristics and metaheuristics for the improvement phase, and evaluating over larger problem instances.

References

[1]. Luss H. Equitable Resource Allocation. John Wiley & Sons, Inc., 2012.

[2]. Vanderster D. C. Resource Allocation and Scheduling Strategies Using Utility and the Knapsack Problem on Computational Grids. PhD thesis, Victoria, B.C., Canada, 2008.

[3]. Pinel F., Dorronsoro B., Bouvry, and Khan S. Savant: Automatic parallelization of a scheduling heuristic with machine learning. In Proc. of the World Congress on Nature and Biologically Inspired Computing, 2013, pp. 52–57.

[4]. Treffert D. The savant syndrome: an extraordinary condition. A synopsis: past, present, future. Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 364, no. 1522, 2009, pp.1351–1357.

[5]. Treffert D. Extraordinary People: Understanding Savant Syndrome. iUniverse, 2006.

[6]. Bouvet L., Donnadieu S., Valoptdois S., Caron C., Dawson M., and Mottron L. Veridical mapping in savant abilities, absolute pitch, and synesthesia: an autism case study. Frontiers in Psychology, vol. 5, no.106, 2014.

[7]. Pinel F., Dorronsoro B., and Bouvry P. The virtual savant: Automatic generation of parallel solvers. Information Sciences, vol. 432, 2018, pp. 411–430.

[8]. Massobrio R., Dorronsoro B., Palomo-Lozano F., Nesmachnow S., and Pinel F. Generación automática de programas: Savant Virtual para el problema de la mochila. In XI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, pages 1–10, 2016.

[9]. Massobrio R., Dorronsoro B., Nesmachnow S., and Palomo-Lozano F. Automatic program generation: Virtual savant for the knapsack problem. In Proc. of the International Workshop on Optimization and Learning, 2018, pp. 1–2

[10]. Kellerer H., Pferschy U., and Pisinger D. Knapsack Problems. Springer, 2004.

[11]. Pinel F. and Dorronsoro B. Savant: Automatic Generation of a Parallel Scheduling Heuristic for Map-Reduce. International Journal of Hybrid Intelligent Systems, vol. 11, no. 4, 2014, pp. 287–302.

[12]. Nemhauser G. L. and Ullmann Z. Discrete dynamic programming and capital allocation. Management Science, vol. 15, no. 9, 1969, pp. 494–505.

[13]. Harman M., Krinke J., Medina-Bulo I., Palomo F., Ren J., and Yoo S.. Exact scalable sensitivity analysis for the next release problem. ACM Transactions on Software Engineering and Methodology, vol. 23, no. 2, 2014, pp. 1–31.

[14]. Bagnall A., Rayward-Smith V., and Whittley I. The next release problem. Information and Software Technology, vol. 43, no. 14, 2001, pp. 883–890.

[15]. Vinyals O., Fortunato M., and Jaitly N. Pointer networks. In Advances in Neural Information Processing Systems 28, pp. 2692–2700, 2015.

[16]. Hu H., Zhang X., Yan X., Wang L., and Xu Y. Solving a new 3d bin packing problem with deep reinforcement learning method. CoRR, abs/1708.05930, 2017.

[17]. Dorronsoro B. and Pinel F. Combining machine learning and genetic algorithms to solve the independent tasks scheduling problem. In Proc. of the 3rd IEEE International Conference on Cybernetics, 2017, pp. 1–8.

[18]. Massobrio R., Dorronsoro B., and Nesmachnow S. Virtual savant for the heterogeneous computing scheduling problem. In Proc. of the International Conference on High Performance Computing & Simulation, 2018 , pp. 1–7.

[19]. Chang C.-C. and Lin C.-J. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology, vol. 2, no. 27, 2011, pp. 1–27.

[20]. Massobrio R., Nesmachnow S., and Dorronsoro B. Support Vector Machine Acceleration for Intel Xeon Phi Manycore Processors. In Proc. of the Latin America High Performance Computing Conference, 2017, pp. 1–14.

[21]. Jeffers J., Reinders J., and Sodani A. Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition. Elsevier Science, 2016.

[22]. Rodríguez, S., Parodi, F., and Nesmachnow, S. Parallel evolutionary approaches for game playing and

verification using Intel Xeon Phi. Journal of Parallel and Distributed Computing, 2018. DOI: 10.1016/j.jpdc.2018.07.010

Информация об авторах / Information about authors

Рензо МАССОБРИО является преподавателем и исследователем на инженерном факультете Республиканского университета, Уругвай с 2014 года. Получил степень магистра наук в области компьютерных наук в Республиканском университете. Сфера его научных интересов – вычислительный интеллект, метаэвристика и высокопроизводительные вычисления, применяемые для решения сложных задач оптимизации.

Renzo MASSOBRIO is a teaching and research assistant at the Faculty of Engineering, Universidad de la República, Uruguay since 2014. He holds a degree of M.Sc. in Computer Science both from Universidad de la República. His research interests are computational intelligence, metaheuristics, and high-performance computing applied to solving complex optimization problems.

Бернабе ДОРРОНСОРО ДИАЗ получил степень PhD в университете Малаги, Испания. В настоящее время он является научным сотрудником факультета компьютерных наук и инженерии Кадисского университета. Он удостоен стипендии Рамона-и-Кахала в Кадисском университете, Испания. В число его научных интересов входят автоматическое программирование, многоцелевая оптимизация, машинное обучение, управление ресурсами в центрах данных, облачные вычисления, эволюционные алгоритмы и т.д.

Bernabé DORRONSORO DÍAZ has a Ph.D. degree in Computer Science from University of Málaga. Currently he is a researcher assistant at the Computer Science Engineering Department of the University of Cadiz, Spain. He is the Ramón y Cajal Fellow at University of Cádiz. His research interests include automatic programming, multi- and many-objective optimization, exact approaches, machine learning, resource management for data-centers, cloud computing, sustainable computing, evolutionary algorithms, ets.

Серджо Энрике НЕСМАЧНОВ КАНОВАС обладает степенью PhD в области компьютерных наук, полученной в Республиканском университете, Уругвай. В настоящее время он занимает должность профессора в Вычислительном центре Института компьютерных наук Инженерного факультета Республиканского университета. Основные научные интересы: параллельные и распределенные вычисления, научные вычисления, эволюционные алгоритмы и метаэвристика, а также численные методы.

Sergio Enrique NESMACHNOW CÁNOVAS has a Ph.D. degree in Computer Science from Universidad de la República, Uruguay. He currently holds an Aggregate Professor position in the Numerical Center (CeCal) at Computer Science Institute, Engineering Faculty. His main research interests are parallel and distributed computing, scientific computing, evolutionary algorithms and metaheuristics, and numerical methods.