EPISTEMOLOGÍA E HISTORIA DE LA CIENCIA

SELECCIÓN DE TRABAJOS DE LAS XIV JORNADAS

VOLUMEN 10 (2004), Nº10

Pío García
Patricia Morey
Editores

ÁREA LOGICO-EPISTEMOLÓGICA DE LA ESCUELA DE FILOSOFÍA
CENTRO DE INVESTIGACIONES DE LA FACULTAD DE FILOSOFÍA Y HUMANIDADES
UNIVERSIDAD NACIONAL DE CÓRDOBA

Esta obra está bajo una Licencia Creative Commons atribución NoComercial-SinDerivadas 2.5 Argentina

Hacia una clasificación de las simulaciones computacionales

*Zenón Pereyra / Víctor Rodríguez**

Introducción

Las simulaciones computacionales ocupan un lugar especial dentro de las prácticas científicas contemporáneas. Ellas aparecen en contextos tan diversos como en las investigaciones sobre inteligencia artificial o en tecnologías sofisticadas de la más variada índole. Resulta difícil establecer criterios de ponderación epistemológica de estas prácticas sin un adecuado marco clasificatorio para las mismas. Esta tarea ha sido realizada parcialmente desde numerosos puntos de vista, pero, en nuestra opinión, la mayoría de los enfoques estudiados adolecen de cierto regionalismo que no deja ver claramente matices generales de considerable protagonismo metodológico y conceptual.

En función de una evaluación del estado de esta cuestión, proponemos un marco esquemático basado en una antigua propuesta esbozada por P. Samuelson para las ciencias económicas, que pretendemos adaptar al sector de investigación bajo análisis. Estimamos que el enfoque adoptado contribuye de una manera programática a articular otro modo de indagación en áreas relacionadas con la inteligencia artificial y las ciencias de la computación. En este sentido, estimamos que expresa una alternativa taxonómica.

A los efectos de dar un marco a la organización propuesta, comenzaremos por exponer algunos ejemplos de programas computacionales que representan estilos relevantes de programación de simulaciones. La lista que se expone a continuación está lejos de ser abarcadora de las múltiples tendencias que existen actualmente en este activísimo sector de investigación. Sólo pretende reflejar un conjunto básico de ellas, pero suficientemente representativo como para ilustrar algunos aspectos relevantes de la arquitectura programática en uso corriente. Como sucede con cualquier muestreo, en el mejor de los casos sólo se presentan indicadores de tendencias y estilos de investigación y desarrollo de programas computacionales. Por este motivo, las conclusiones que extraemos son provisorias y están sujetas a indagaciones y estudios posteriores de otros ejemplos importantes. No obstante, hemos considerado que este trabajo de campo y las consideraciones que lo acompañan justifican su presentación en el ámbito de la filosofía de la ciencia.

Los programas analizados por nuestro proyecto de investigación hasta la fecha pueden ser ordenados en torno de:

a) Simulaciones realizadas en supercomputadoras.

b) Programas de procesos evolutivos y sus variantes, que van desde programas con paradigmas darwinianos hasta algoritmos genéticos.

c) Programas *standard*, asociados a simulaciones de muchas áreas de investigación científica y llevados a cabo por medio de computadoras de nivel intermedio.

* CIFFyH. Universidad Nacional de Córdoba.

d) Programas asociados con lo que suele llamarse filosofía computacional de la ciencia, que en buena medida se superponen con instancias de las clasificaciones anteriores. Un caso particular de ellos, es el de los programas que simulan descubrimientos científicos.

e) Implementaciones computacionales en dispositivos de distinta naturaleza, como es el caso del ADN computacional y su incidencia en los modelos matemáticos.

Dado que las exploraciones no se han realizado de un modo sistemático, sino atendiendo principalmente a las disponibilidades de acceso a software especializado vinculado a departamentos de investigación en estas áreas, resulta difícil obtener una adecuada taxonomía de todas las entradas sugeridas. Esta es una de las razones por las cuales lo expuesto aquí configura un enfoque programático.

En la búsqueda de un marco suficientemente general y luego de evaluar diferentes opciones en uso en las series de publicaciones de ciencias de la computación, como las *Lectures Notes in Computer Science*, nos ha parecido que el enfoque de Samuelson presenta una caracterización robusta y a la vez general, y puede ser adaptado para una primera aproximación al conjunto de programas disponibles.

En la selección de ejemplos se ha tenido en cuenta un objetivo de nuestra línea de investigación, que es el planteamiento y resolución de problemas y sus implementaciones computacionales. Por ello no se ha prestado especial atención a la gama de programas tipo "crunching numbers", esto es, devoradores de números, más propios de las super-computadoras y técnicas de simulación asociadas. Aunque sin estar inmerso en la inteligencia artificial, el enfoque de solucionar automáticamente problemas es central para ella, para el aprendizaje automatizado, y para toda un área comprendida dentro de lo que Turing llamó "inteligencia de la máquina"[1]. Recordemos que en la década de 1950 "inteligencia artificial" se asociaba al objetivo de lograr que las máquinas exhibieran un comportamiento similar al humano[2], lo que supondría el uso de algún concepto adecuado de inteligencia.

Este concepto fue progresivamente divergiendo de su significado original, para estar actualmente asociado con metodologías particulares que tienen un objetivo distinto, como el que las computadoras resuelvan automáticamente problemas; como lo ha demostrado, por ejemplo, el campo de investigación llamado *Programación Genética*, donde la computadora es capaz de resolver ciertos problemas en forma automática especial.

Los programas computacionales para la resolución de problemas en inteligencia artificial, específicamente de aprendizaje automatizado, son simulaciones computacionales que utilizan modelos en los que se crea un universo artificial, el que en forma explícita o implícita es definido por el programa. Como hace Ziegler[3], podemos visualizar el modelado y la simulación desde una simple y elegante perspectiva: el modelado es el puente entre un sistema real y un modelo, mientras que la simulación es la conexión a un dispositivo computacional que puede activar mecanismos dentro del modelo. Estos programas pueden ser construidos a partir de diversos marcos teóricos o lenguajes de programación, cuyos componentes son una colección integral de representaciones, funciones y méto-

dos según su arquitectura. A estos objetos se los puede identificar por el rol específico que cumplen dentro de la aplicación.

Como un primer paso hacia la comprensión de un conjunto cualquiera de fenómenos asociados con estos programas, debemos singularizar qué tipo de cosas existen en el conjunto y a partir de allí desarrollar una taxonomía[4]. Esta construcción generalmente utiliza como propia la denominación de algún método o algoritmo que eventualmente utiliza y que puede, en cierta manera, caracterizarlo. También puede servirse de alguna parte saliente u objeto existente en la aplicación, sin que estos constituyan realmente una cortadura conceptual.

Antes de proponer la utilización de la taxonomía de Samuelson[5] para este tipo de aplicaciones, ya común en la clasificación de otros sistemas dinámicos, conviene describir en líneas generales las partes de un programa computacional de aprendizaje automatizado.

Supongamos que el papel protagónico de la aplicación corresponde a lo que se suele denominar "agente". Trataremos de explicar por qué su papel es protagónico. Desde hace tiempo se ha venido desarrollando el concepto de agente, el que algunas veces es definido explícitamente por el programa computacional, según una serie de definiciones a la que se ajusta esa aplicación, como se puede ver en el trabajo de Franklin y Gaesser[6]. En nuestro caso nos serviremos de una caracterización tomada de Russell y Norvig[7]: *"Un agente es algo que puede ser visto como un objeto que percibe su ambiente a través de sensores y actúa contra el ambiente a través de efectores"* y que *"para cada secuencia perceptiva posible, un agente racional ideal deberá efectuar la acción capaz de maximizar su medida de rendimiento, sobre la base de la evidencia provista por la secuencia perceptiva y teniendo en cuenta el conocimiento que el agente debe disponer"* (traducción nuestra).

Según esta definición, un programa como proceso, es decir que transcurre en el tiempo, pone en correspondencia las secuencias perceptivas y las acciones. Las reglas de arquitectura para la construcción de este agente contemplan la definición de elementos destinados a la organización material y lógica, el control de los receptores, la provisión de las secuencias perceptivas y la ejecución de los programas y las acciones.

En resumen, la estructura de un agente estará definida por:

Agente = arquitectura + programa

En un programa de aprendizaje automatizado es necesario distinguir otros elementos cuyo control es responsabilidad directa del agente, como el conjunto de objetos que interactúan entre sí durante el proceso y el conjunto destinado a ordenar su terminación. Dentro del primer conjunto se encuentra lo que denominaremos motor de inferencias: son los elementos que trabajan con una serie de comandos conocidos como reglas de producción, combinando condiciones con acciones o premisas con conclusiones. Utiliza algoritmos independientes del dominio y estructuras de control existentes en la base de conocimientos para solucionar problemas. Este motor puede utilizar inferencias deductivas, inductivas, abductivas o estocásticas. A este conjunto de objetos se agrega la base de conocimientos, la que mantiene datos, información y suposiciones específicas del domi-

nio y representaciones de hechos del mundo real codificados como expresiones en un lenguaje de representación.

Finalmente, en este universo artificial, donde, como dijimos, la acción protagónica corresponde al agente, falta definir lo que identificaremos como una forma de razonamiento, necesaria tanto para formular nuevas inferencias, como para adquirir información de la base de conocimientos o decidir si se continúa o no con el proceso. Se ha interpretado muchas veces que estas formas de razonamiento se ajustan a lo que Pierce distinguió como "deductivo", "inductivo" o "abductivo", pero no es nuestro objetivo presente ir en esa dirección de análisis. En los libros de inteligencia artificial y material relacionado, estos temas suelen ser tratados de modo esquemático y se ajustan a los objetivos de dar un encuadre a razonamientos procedimentales dentro de los marcos lógicos habituales. En este contexto es frecuente asociar la abducción con el proceso de encontrar la mejor explicación para un conjunto de observaciones, o inferir causas a partir de efectos. En general, un modelo para una aplicación de aprendizaje automatizado implementa un proceso inductivo, que tiene como entrada datos o comandos y como salida inferencias o resultados. Dentro de este proceso es posible introducir los conceptos de incertidumbre y de inferencia probabilística. "Incertidumbre" se refiere a la falta de una correcta y adecuada información para utilizar en la toma de decisiones. De esta manera, el razonamiento o inferencia bajo incertidumbre pasa a ser un tema principal en el diseño de modelos de sistemas de aprendizaje automatizado.

Esquemáticamente, se suelen distinguir los tipos de inferencias utilizados por un programa para mantener el proceso como perteneciente a alguno de los siguientes tipos, con su caracterización:

Inferencia deductiva: Regla + caso = resultado.

Inferencia inductiva: Caso + resultado = regla.

Inferencia abductiva: Resultado + regla = caso.

Inferencia estocástica: Procesos estocásticos.

Dada la descripción del protagonista de este universo artificial y de algunas de sus funciones, es necesario considerar los otros tres componentes que eventualmente integran un programa de aprendizaje automatizado y que de algún modo le proveen las definiciones que necesita. Estimamos que M. Bunge[8] aporta un buen marco para nuestros fines. Una teoría general de fondo constituye una parte integral de un modelo[9], que para nuestro caso denominaremos "dominio", en el cual tiene lugar la existencia del problema, y en el que también podremos distinguir dos partes: a) la teoría general que lo sustenta, y b) una descripción especial para los objetos o del sistema propuesto. El dominio, que puede ser continuo o discreto, reúne las definiciones de existencia de los objetos dentro del universo simulado, por ejemplo los valores posibles que las variables pueden tomar o la oportunidad de aplicación de determinados métodos.

Una segunda clase de objetos se suele denominar "ambiente". Éste puede ser definido por un estado inicial y por ciertas funciones destinadas a observar y ana-

lizar las relaciones entre agente y el medio en el que opera; por ejemplo, la percepción y acción del agente contra el ambiente, para posteriormente evaluar su actualización. Algunos tipos de ambiente pueden caracterizarse por ser:

> Accesibles o inaccesibles. Determinísticos o no determinísticos. Episódicos o no episódicos. Estáticos o dinámicos. Discretos o continuos

Una tercera clase de objetos, los algoritmos, está constituida por herramientas computacionales que utiliza el agente y pueden ser o no independientes del dominio. Las podemos definir dentro del motor de inferencias y están destinadas a la manipulación de los objetos que constituyen la base de conocimientos a los que puede acceder el agente, a la interpretación del lenguaje de representación del conocimiento, a la comunicación entre objetos, a la provisión de una interfaz con el mundo exterior y, finalmente, a computar las condiciones necesarias y suficientes para finalizar la simulación.

Las características de cada uno de estos elementos pueden ser utilizadas para definir un criterio de cortadura, y de hecho esto es lo que ha sido usual. De esta manera, la taxonomía resultante puede corresponder solamente con una metodología, con la teoría utilizada para definir su dominio, o con alguna característica parcial del programa, eventualmente distante del verdadero objetivo del programa.

Proponemos la clasificación propuesta por Samuelson para sistemas dinámicos para utilizarla con programas de aprendizaje automatizado, suponiendo que se trata de procesos dinámicos y en los que una simulación imita un proceso por otro proceso[10]. En este contexto se intenta simular sistemas que tienen una dimensión temporal, y como consecuencia de ello, tanto el modelo como la simulación son dinámicos.

Esta taxonomía contempla las siguientes clases:

1. Estático y estacionario.

2. Estático e histórico.

3. Dinámico y causal.

4. Dinámico e histórico.

5. Estocástico y no histórico.

6. Estocástico e histórico.

En lo que sigue vamos a tratar de ejemplificar la utilización de esta taxonomía en la clasificación de aplicaciones de aprendizaje automatizado y sistemas afines. Conviene hacer notar que interpretamos que la caracterización de Samuelson referente a modelos históricos o no históricos se corresponde a las clases de programas que podemos definir como sistemas supervisados y no supervisados.

Para un modelo histórico, el comportamiento en algún período de tiempo futuro dependerá no solamente de las condiciones iniciales dentro del dominio, sino también de las condiciones y perturbaciones iniciales exógenas. La existencia de esta variable exógena, controlable desde el exterior del sistema, hace que este

tipo de sistemas se incluya en alguna de las categorías de sistemas históricos de Samuelson.

Por otra parte, se definen como sistemas supervisados aquellos en los que, en cualquier situación, es posible percibir ambas, la entrada y salida de un componente. En este caso, las salidas pueden modificarse por un instructor externo, la variable exógena, la que a su vez también puede proveer nuevas entradas, supervisando de esta manera el proceso. Con esta condición, es posible reducir dentro de la taxonomía de Samuelson el número de tipos de modelo de simulación a solamente tres, ya que su historicidad o no historicidad estará determinada por la existencia, o no, de alguna variable exógena; o lo que es lo mismo, se puede especificar si se trata de un sistema con aprendizaje supervisado o no supervisado.

Modelos estáticos

Se encuentran en esta categoría sistemas incrementales o no incrementales, supervisados o no supervisados, cuyo motor de inferencias produce inferencias deductivas o inductivas.

Un primer ejemplo es el programa AQ[11], esto es un sistema que aprende reglas a partir de árboles de decisión, que son utilizados deductivamente.

El programa Cobweb[12] es una implementación simple del algoritmo de acumulación conceptual incremental *COBWEB* de Douglas H. Fisher[13], que trata de sintetizar los conceptos existentes en anteriores sistemas de aprendizaje automatizado y ciertos resultados derivados de la psicología cognitiva. El mismo fue inspirado inicialmente por la investigación de efectos de nivel básico.

Los programas *ID3* (Quinlan, 1993), *ID4, ID5,* son una serie de implementaciones que comienzan con el bien conocido algoritmo *ID3*[14] (el que no puede ser considerado incremental todavía, dado que no conserva la memoria), para aprendizaje a partir de ejemplos y que utiliza una representación en árbol para conceptos15. Produce árboles de decisión que discriminan instancias positivas y negativas, las que son representadas por simples vectores con atributos nominales representados a su vez por listas ordenadas. *ID4* e *ID5,* también son sistemas supervisados evolucionados, y son incrementales.

Completaremos los ejemplos de modelos estáticos con el programa *VERSION-SPACE16.* Hemos trabajado con una implementación básica de *Version Space,* algoritmo para simular casos de aprendizaje incremental a partir de ejemplos. Esta función toma una lista de ejemplos, donde un ejemplo es una lista cuyo primer elemento es un signo ' + ' o uno ' - ', para indicar un ejemplo positivo o uno negativo, y cuyo segundo elemento es una descripción de ese ejemplo (o instancia de él). Utiliza el algoritmo *Version Space* para determinar los conjuntos de las generalizaciones más específicas y más generales, consistentes con todos los ejemplos. Una propiedad importante de este algoritmo es que se puede definir como verdaderamente incremental, es decir que no es necesario volver atrás para reexaminar los ejemplos anteriores.

Modelos causales

Un ejemplo de modelo causal lo constituye el programa de aprendizaje automatizado ACCEL17. Se trata de un sistema de aprendizaje inductivo orientado a ser independiente del dominio que utiliza axiomas cláusulas de Horn de primer or-

den con un algoritmo de abducción, cuyos resultados se obtienen a partir de inferencias abductivas. Como demostración, tiene implementados cinco tipos de problemas, que se resuelven satisfactoriamente, en los que propone como posibles tres métodos distintos. 1) Entendimiento de textos o diálogos (*plan recognition*). Puede simular dos situaciones: en la primera un lector infiere objetivos y planes de alto nivel a partir de los caracteres de un texto para explicar los eventos y acciones descriptos en el texto, y en la segunda el entendimiento de diálogos, en el cual un participante infiere objetivos y planes de otros participantes basados en las expresiones intercambiadas en una conversación. 2) Diagnóstico médico. Implementa un modelo denominado "Diagnóstico basado en el cubrimiento de un conjunto" (Generalized Set Covering - GSC - model of diagnosis), proporcionando resultados empíricos a partir del diagnóstico de casos (pacientes). Usa una base de conocimientos bastante grande con aproximadamente seiscientas reglas. 3) Diagnósticos basados en modelos. Utiliza abducción para efectuar diagnósticos basados en modelos, en este caso simula un circuito lógico, más un par de sistemas dinámicos para proporcionar los datos y conocimientos necesarios acerca de la correcta estructura y comportamiento de los respectivos sistemas, infiriendo fallas a partir de los primeros.

Otro ejemplo de modelo causal es el programa LAB18. Induce reglas abductivas de la forma causa -> efecto, a partir de un conjunto de ejemplos, para luego utilizarlos abductivamente.

Modelos estocásticos

En el programa lil-gp19 y su implementación para sistemas Windows *"Ensayo de Programación Genética" (EPG)*20, el agente del programa efectúa inferencias estocásticas utilizando algoritmos genéticos. En ambos casos el objeto agente no está explícitamente definido, pero se lo caracteriza como de "Algoritmos Genéticos", ya que en su arquitectura, el componente característico para ambos programas es el objeto "Algoritmos", que formaría parte del motor de inferencias.

Los algoritmos genéticos son un modelo de aprendizaje automatizado que define su comportamiento a partir de una metáfora del proceso de evolución existente en la naturaleza. Esto se hace creando en una máquina una población de individuos, generados con funciones de simulación de procesos estocásticos, representados por "cromosomas". En esencia un conjunto de cadenas de caracteres que son análogos a los cromosomas de base 4 que nosotros podemos ver en nuestro propio DNA. Los individuos de esta población se utilizan posteriormente en un proceso de evolución.

Entre las numerosas las aplicaciones de modelos estocásticos, nos parece conveniente citar las implementaciones del modelo de Ising que utilizan simulaciones de Monte Carlo e implementan los algoritmos de Metropolis o de Glauber, para dos o tres dimensiones, existentes en diversas versiones y lenguajes de programación, muy populares como problemas de examen en diversas universidades21. Otro conjunto de sistemas estocásticos es la familia de programas "Boids"22, implementados en Internet como applets, que simulan el comportamiento aparentemente al azar de bandadas de pájaros.

Conclusiones:

Estimamos que los ejemplos citados son elocuentes y significativos como ilustración de los marcos propuestos por Samuelson, los que han sido adaptados para nuestros objetivos programáticos de estructurar una base para el desarrollo de una taxonomía en torno de los programas vigentes en el ámbito analizado. Si bien somos concientes de los límites de nuestra propuesta, pareciera ser que ésta es una característica de las estrategias *bottom up*. Naturalmente, esta estrategia puede ser complementada con la opuesta, de arriba hacia abajo, pero hemos preferido exponer sólo una variante de nuestra exploración y ejercicio de clasificación porque estimamos que es la de mayor necesidad dada las exhuberantes variedades de programas existentes. Atendiendo a que solo estamos en presencia de una muestra de rango muy breve, nuestras conclusiones no pueden ser sobredimensionadas. No obstante, esta clasificación promete modos de organización considerablemente originales para explorar, dado que el no se encuentra en la literatura específica un tratamiento que establezca jerarquías similares ni una estratificación de las características dinámicas similar al presentado aquí. Quedan muchas tareas por realizar en esta línea de trabajo, pero la organización de los programas sugiere un relativo avance en materia de criterios taxonómicos.

Notas

1 Turing, A., 1948, "Intelligent Machinery", reporte del Physical Laboratory, Londres. Publicado en 1969 en B. Meltzer y D. Michie (ed.), *Machine Intelligence*, vol. 5, pág. 3-23, Edinburgh University Press. Turing,, A., 1950, "Computing Machinery and Intelligence", *Mind*, 59:433-460.

2 Samuel, A., 1983, "AI: Where It Has Been and Where It Is Going", en *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI 1983)*, Alan Bundy, ed., Karlsruhe, FRG, August 1983. William Kaufmann, 1152-1157.

3 Ziegler, B.P., 1976, *Theory of Modeling and Simulation*, John Wiley and Sons, New York. (Li Min Fu, pág. 310)

4 Simon, H., 1979, *Las ciencias de lo artificial*, Ed. ATE, Barcelona, pág. 118.

5 Samuelson, H., 1947, *Foundations of Economic Analysis*. Cambridge - Harvard Univ Press, pág. 315-317

6 Stan Franklin S. and Gaesser A. 1996, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages*, Springer-Verlag.

7 Russell, S. and Norvig, P., 1995, *Artificial Intelligence: A Modern Approach*. Chapter 2, New York: Prentice-Hall.

8 Bunge, M., 1967, *Scientific Research II*. Springer - Verlag, Berlin. Cfr. Bunge, M., 1969, "Analogy, Simulation, Representation», *Revue Internationale de Philosophie*, 87, pág. 16-33.

9 Hartmann, S. ,1996, "The World as a Process" en *Modeling and Simulation in the Social Sciences from the Philosophy of Science Point of View*, R. Hegselmann, U Mueller and K.G. Troitzsch, ed., Kluwer Academic Publ., Dordrecht, pág. 82.

10 Hartmann S., 1996, op.cit.

11 Dietterich, T.G. & Michalski, R.S., "Learning to Predict Sequences" en *Machine Learning - An Artificial Intelligence Approach*, Vol. 2, pág. 63, Morgan Kaumann Publisher, Inc., Los Altos, Ca, 1986.

12 Tenemos dos versiones extraídas del repositorio de la Carnegie Mellon University:

1) en Lisp (areas/learning/systems/utexas/code/ml_code.tgz) y

2) en Prolog (areas/learning/systems/learn_pl/cobweb/).

13 Fisher, D.H., "Knowledge Acquisition Via Incremental Conceptual Clustering", *Machine Learning*, 2 , pags. 139-172, 1987

14 Quinlan, J.R., 1986,"Induction of decision trees", *Machine Learning*, 1, pags. 81-106.

15 Quinlan, J.R., 1983, "Learning efficient classification procedures and their application to chess and games", R.S. Michalski, J.G.Carbonell and T.M. Mitchell (Eds.), *Machine Learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kauffmann.

16 Russell, S. & Norvig, op. cit. pág. 549.

17 Ng, H.T., 1992, "A General Abductive System with Application to Plan Recognition and Diagnosis", AI 92-177, Artificial Intelligence Laboratory, The University of Texas at Austin.

18 Artificial Intelligence Laboratory, The University of Texas at Austin.

19 Lil-gp es una realización colectiva desarrollada originalmente por Douglas Zongker, Bill Punch, que cuenta además con la colaboración de Robert J. Steele y Jan Jihn-Fa de la Michigan State University, que forman el Genetic Algorithms Research and Applications Group (GARAGe), en constante evolución.

20 EPG es una aplicación desarrollada en la Universidad de Córdoba, España.

21 Por ejemplo el grupo de Theoretical Physics, Department of Physics Royal Institute of Technology, Estocolmo, Suecia. http://courses.physics.kth.se/5A1354/

22 Reynolds, C.W., 1986, "Boids", http://www.red3d.com/cwr/boids/